國立臺灣大學電機資訊學院電機工程學系

碩士論文

Department of Electrical Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

基於優化無跡卡爾曼濾波器與迭代式 LQR 追蹤之無標記式單人三維人體動作估測系統

A Markerless Multi-view 3D Human Motion Estimation System for Single Person with Modified Unscented Kalman Filter and iterative LQR tracking

賴橋

Chiao Lai

指導教授：連豊力 博士

Advisor: Feng-Li Lian, Ph.D.

中華民國 111 年 6 月

June 2022

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

國立臺灣大學碩士學位論文
口試委員會審定書

MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

基於優化無跡卡爾曼濾波器與迭代式 LQR 追蹤之無標記式單人三維人體動作估測系統

A Markerless Multi-view 3D Human Motion Estimation System for Single Person with Modified Unscented Kalman Filter and iterative LQR tracking

本論文係___賴橋___(姓名)__R09921005__（學號）在國立臺灣大學_電機工程學系_(系/所/學位學程)完成之碩士學位論文，於民國_111_年_6_月_30_日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department / Institute of _____Electrical Engineering_____ on 30 (date) June (month) 2022 (year) have examined a Master's thesis entitled above presented by Chiao Lai (name) _____R09921005_____ (student ID) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

連豐力		吳沛遠
連豐力（指導教授）	吳育任	吳沛遠

系主任 吳忠幟: _____

doi:10.6342/NTU202202468

誌謝

碩士生涯即將結束，回首這兩年的過往，一路上充滿挑戰，在多位老師與實驗室的幫助下，這篇碩士論文逐漸的成形。

首先要感謝**連豐力**教授，作為指導教授，在每週的週報中訓練了我們養成記錄每天工作的習慣，除了在研究時可以隨時審視自己過去做了什麼嘗試、還有哪些改進方向外，也能用以保護自身的研究成果，這對於想在研究這條路走下去的人來說是一個很重要的習慣。此外，在文獻整理、數據呈現與分析到研究價值的呈現，這些都是我在實驗室中才學到得來不易的知識。老師在研究方面不吝給予的支持與建議也一直是我研究能如此順利的原因。

感謝**吳昱任**教授，作為此研究計畫的計畫主持人，幫我們整合了許多來自不同實驗室的資源，讓我們的實驗能從實驗室到真實職棒球場，提升了數據的完整性與真實度。感謝有這個計畫，我才能真實地體會到參與一個大型的研究計劃是什麼樣的體驗。也感謝**吳沛遠**教授、**徐瑋勵**教授與**黃致豪**教授，整個計畫的每一個部分都是各個實驗室心血的結晶，多虧您們的協助，本計畫才能產出不錯的結果並順利地結案。

感謝 NCSLab **與計畫合作實驗室**的夥伴們。感謝**柏宇**學長與**彥穎**學長，沒有兩位在前兩年的付出，我在開始研究時就沒有這些資料供我做初步的分析，研究方向也沒辦法如此快速的定下。感謝**國郡**、**柏佑**及**王捷**學長，從進實驗室起你們一直是我觀摩的對象，對期刊論文報告的認真整理與分析是我這兩年學到最多的地方之一。感謝**仲宇**、**佳芸**、**宇強**、**俊學**，兩年的研究之路，多虧有你們才不顯得沉悶，彼此互相交流的意見，也讓我這個除了研究不大動腦的人能夠順利畢業。也感謝**之蕙**，在協助你研究的過程中我才體會到了教學相長的感覺。感謝**其他學弟**，在閒暇之餘能與我交流研究上的看法，這樣的交流過程是我覺得在實驗室中最寶貴的收穫。

最後感謝我**父母與家人**，感謝您們在我求學的這十八年來，無怨無悔的付出，永遠做我經濟上最堅強的後盾，使我在求學的過程中不用為金錢的問題而煩惱。

賴橋 謹誌

中華民國一百一拾一年八月七日

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

基於優化無跡卡爾曼濾波器與迭代式 LQR 追蹤之無標記式單人三維人體動作估測系統

研究生：賴橋　　指導教授：連豊力 博士

國立臺灣大學　電機工程學系

摘要

近幾年來，三維人體動作估測一直都是個熱門的研究主題。從電影產業、復健治療到運動分析，越來越多的應用環境使得人們對三維人體動作估測的精準度與便利性有了更高的要求。隨者深度學習的興起，漸漸地有許多無標記式的估測方法被提出。但這些方法通常都會遇到缺乏室外三維標記資料的問題，使得提出的方法在現實情境中沒有辦法得到如預期的結果。

為了避開這個問題，一個僅基於二維人體關節偵測的方法在此篇論文中被提出。考量到直接將二維的偵測結果做三維重建可能會使得三維出測結果出現巨大的誤差，此三維重建結果還會經過骨架優化的步驟。此骨架優化由兩部分組成。第一部分為基於骨架模型的關節角度估計。第二部分則為動作平滑化。在關節角度估計中，除了關節角度的計算，來自三維重建的異常關節也會在被提出的異常分量排

doi:10.6342/NTU202202468

除無跡卡爾曼濾波器濾除以達到提升估計強韌性的目的。在動作平滑化中，除了位置之外，速度與加速度準確度這些高階次的指標也會在這一步得到顯著的提升。

最後，透過模擬與實驗，數據化的驗證所提出方法的效果與性能，以證明其可行性與精確度。

關鍵字：

人體動作估測、人體姿態估測、異常分量濾除、動作平滑化

doi:10.6342/NTU202202468

A Markerless Multi-view 3D Human Motion Estimation System for Single Person with Modified Unscented Kalman Filter and iterative LQR tracking

Student: Chiao Lai Advisor: Dr. Feng-Li Lian

Department of Electrical Engineering

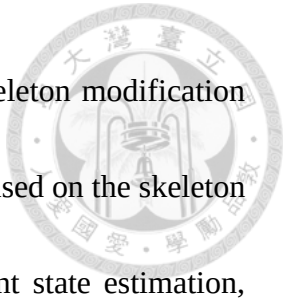
National Taiwan University

ABSTRACT

In recent years, 3D human motion estimation has been a popular research topic. From the film industry, rehabilitation therapy to sports analysis, more and more application environments make people require higher accuracy and convenience of 3D human motion estimation. With the rise of deep learning, many markless estimation methods have been proposed. However, those methods usually encounter the problem of a lack of outdoor labeled data so that the estimation results are not as good as expected in real-world situations.

To avoid this problem, a method based only on 2D human keypoint detection is proposed in this thesis. Considering that direct 3D reconstruction of the 2D detection results may cause huge errors in the 3D estimation results, the 3D reconstruction results

doi:10.6342/NTU202202468



will also undergo the 3D skeleton modification process. The 3D skeleton modification process consists of two parts. The first part is joint state estimation based on the skeleton kinematic model. The second part is motion smoothing. In the joint state estimation, besides the calculation of the joint angle, the outlier keypoint from the 3D reconstruction will also be filtered out with the proposed outlier-component rejecting UKF (OCR-UKF) to improve the robustness of the estimation. In motion smoothing, in addition to position, higher-order metrics such as velocity and acceleration accuracy will also be significantly improved in this step.

Finally, through simulation and experiment, the properties and performance of the proposed method are verified with data to prove its feasibility and accuracy.

Keywords:

Human motion estimation, human pose estimation, outlier component rejection, motion smoothing

doi:10.6342/NTU202202468

CONTENTS

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

LIST OF FIGURES

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

LIST OF TABLES

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

Chapter 1
Introduction

In this chapter, the motivation of this thesis is discussed in Section 1.1. Then the problem formulation for this thesis is provided in Section 1.2. Section 1.3 states the contributions of this thesis. Finally, the organization for the rest chapters of this thesis is provided in Section 1.4.

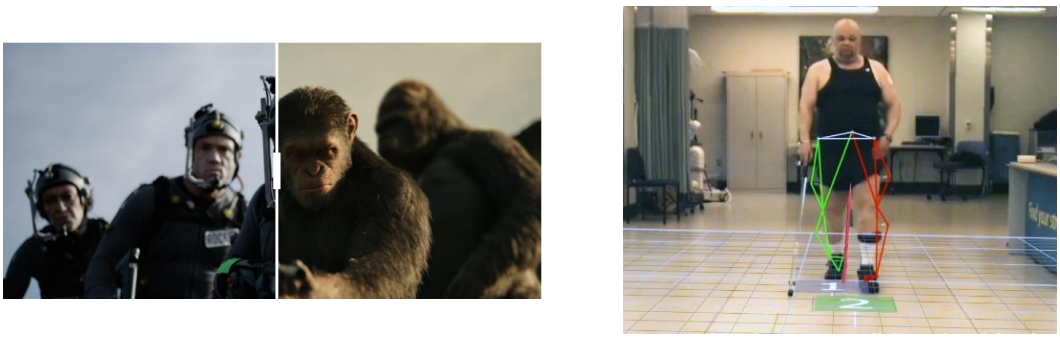
1.1 Motivation

In recent years, 3D human motion estimation is a widely used technology in several fields. In the film industry, there have been several movies using optical motion capture system to capture the accurate motion of the actors/actresses. With accurate motions and computer graphic technology, the filmmakers can make unreal creatures and have them act like they are real and emotional. A famous film, “War for the Planet of the Apes”, used this technology to do so [75: Bishop 2017]. The apes in this movie were actually acted by real actors and actresses in labs with high-precision optical motion capture systems as shown in Figure 1.1 (a).

Not only for films, nowadays, 3D human motion estimation is also used in hospitals, as shown in Figure 1.1 (b). The rehabilitation of stroke or traumatic brain injury patients is a tough and long process. The treatments highly rely on the experiences of physical

doi:10.6342/NTU202202468

therapists. As [76: Hogan 2020] mentioned, with the 3D motion estimation, physical therapists can analyze the movements of patients with mobility-limiting conditions such as Parkinson's disease. This would provide the ability to characterize the abnormal motion with high accuracy, and it's necessary for surgical evaluations.



(a) (b)

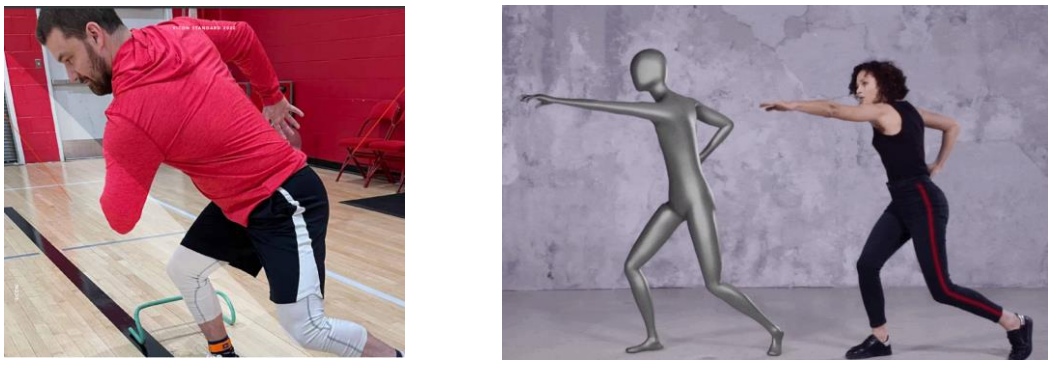
Figure 1.1: The applications for 3D human motion technology with optical motion capture system.
(a) Motion transferring in film industry [75: Bishop 2017] (b) rehabilitation [76: Hogan 2020]

However, to use the high-accuracy optical motion capture systems, the scenario environment should be an indoor and stable environment like labs. Also, the expansive cost of the whole optical motion capture system is one of the weaknesses to increase its universality. Therefore, there are some wearable devices developed for issues. [77: VICON 2022] shows that wearable inertia measurement units (IMUs) can help athletes to monitor their motions and furthermore give the clinicians a signal whether the anterior cruciate ligament (ACL) injuries occur at both prevention and rehabilitation stages as shown in Figure 1.2 (a).

doi:10.6342/NTU202202468

Despite the fact that the wearable devices are cheap and don't need abundant room to implement, not all the cases suit the wearable devices. For some sports, like baseball, the players can't wear any devices or sensors during the official games. In these situations, some vision-based 3D human motion estimation algorithms are the most suitable methods.

In the last few years, due to the growth of computational power and the machine learning technique, more and more problems that used to be regarded as almost unsolvable can be modeled with artificial neural networks. Vision-based human motion estimation is one of them. In Figure 1.2 (b), [78: Dutt 2018] shows a good example that the dancing motions of humans can be captured and reproduced with only videos. With these techniques, 3D human motion estimation becomes a universal and easy-to-implement technology.

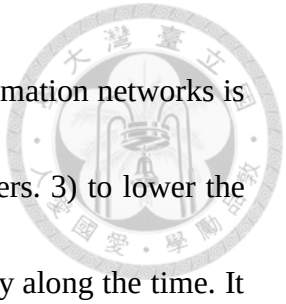


(a) (b)

Figure 1.2: The applications for 3D human motion technology with other measurement techniques.
(a) IMU-based motion estimation [77: VICON 2022] (b) vision-based motion estimation [78: Dutt 2018]

But there are still some challenges for vision-based human motion estimation. 1) the precision of vision-based human motion estimation is lower than the optical motion

doi:10.6342/NTU202202468



capture systems. 2) the computational cost for 3D human motion estimation networks is still high for personal computers. Usually, the networks run on servers. 3) to lower the computational loading, few algorithms consider the motion continuity along the time. It also causes bad velocity and acceleration estimation, which we don't want especially for athlete performance monitoring. 4) since the train data for 3D human motion estimation models are recorded in a stable environment, the high noise of real-world data often causes bad estimation. Sometimes, the estimated results don't even conform to real human-like motions.

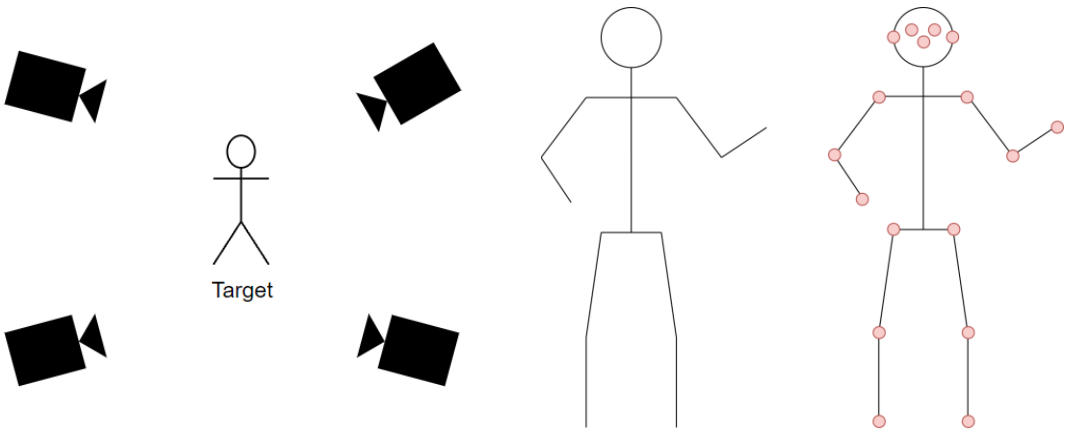
Considering the problems above, we want to propose a method that combines the advantages of pattern recognition for machine learning and model-based method for signal processing. Therefore, the estimated motions would be robust to the real-world noise and also conform to the human motion model.

1.2 Problem Formulation

For vision-based multi-view 3D human motion estimation, the estimated target would make actions under a multi-camera system, which is illustrated in Figure 1.3 (a). The purpose is to estimate the motions, including position, velocity, and acceleration, of each keypoint on a human body. The keypoints for human bodies are shown on Figure 1.3 (b). There are totally 17 keypoints on a human body, including nose (Nose), left eye (LEye), right eye (REye), left ear (LEar), right ear (REar), left shoulder (LShoulder), right

doi:10.6342/NTU202202468

shoulder (RShoulder), left elbow (LElbow), right elbow (RElbow), left wrist (LWrist), right wrist (RWrist), left hip (LHip), right hip (RHip), left knee (LKnee), right knee (RKnee), left ankle (LAnkle), right ankle (RAnkle), which is the same definition in Microsoft COCO keypoint detection task [51: Lin et al. 2015].



(a) (b)

Figure 1.3: Illustration on the formulated problems.
(a) the scheme diagram for 3D motion estimation with multi-view systems
(b) the keypoints to estimate the motion for a human body

Based on the times of tracking, the 3D human motion estimation can be divided into single-track scenarios and multi-track scenarios.

1.2.1 Single-track 3D human motion estimation

For single-track 3D human motion estimation, the input data only contain one sequence of human motion. Without any additional information, it needs to estimate the target's 3D motion for each keypoint from the videos of the multi-view system and also resist the huge noise in the real-world images.

doi:10.6342/NTU202202468

1.2.2 Multi-track 3D human motion estimation

For multi-track 3D human motion estimation, the input data contain multiple sequences from the same person with similar motions. This situation is common for repeated actions such as pitching, batting, walking… etc. Those actions have a fixed pattern and few variations in different tracks. In this task, we can leverage the repeatability of motions to improve the robustness of the estimated result and resist the captured noise. Moreover, it can increase the precision of the estimated results.

1.3 Contributions

The master thesis is devoted to the robustness issue of real-world 3D human motion estimation. To fulfill a robust 3D human motion estimation, we divide the whole process into two parts: 1) 3D raw skeleton reconstruction and 2) single-track 3D skeleton stabilization, which is also illustrated in Figure 1.4. The main contribution of this thesis mainly lies in the second part.

In terms of robustness for 3D human motion estimation, this work provides an outlier rejection method to avoid the outliers which cause by the capture noises corrupting the estimated outputs in the second part.

In addition, to consider the estimation of velocity and acceleration, this work implements the iterative LQR motion smoother. With this process, the output motions

doi:10.6342/NTU202202468

have great improvement in velocity and acceleration improvement, which few methods so far consider.

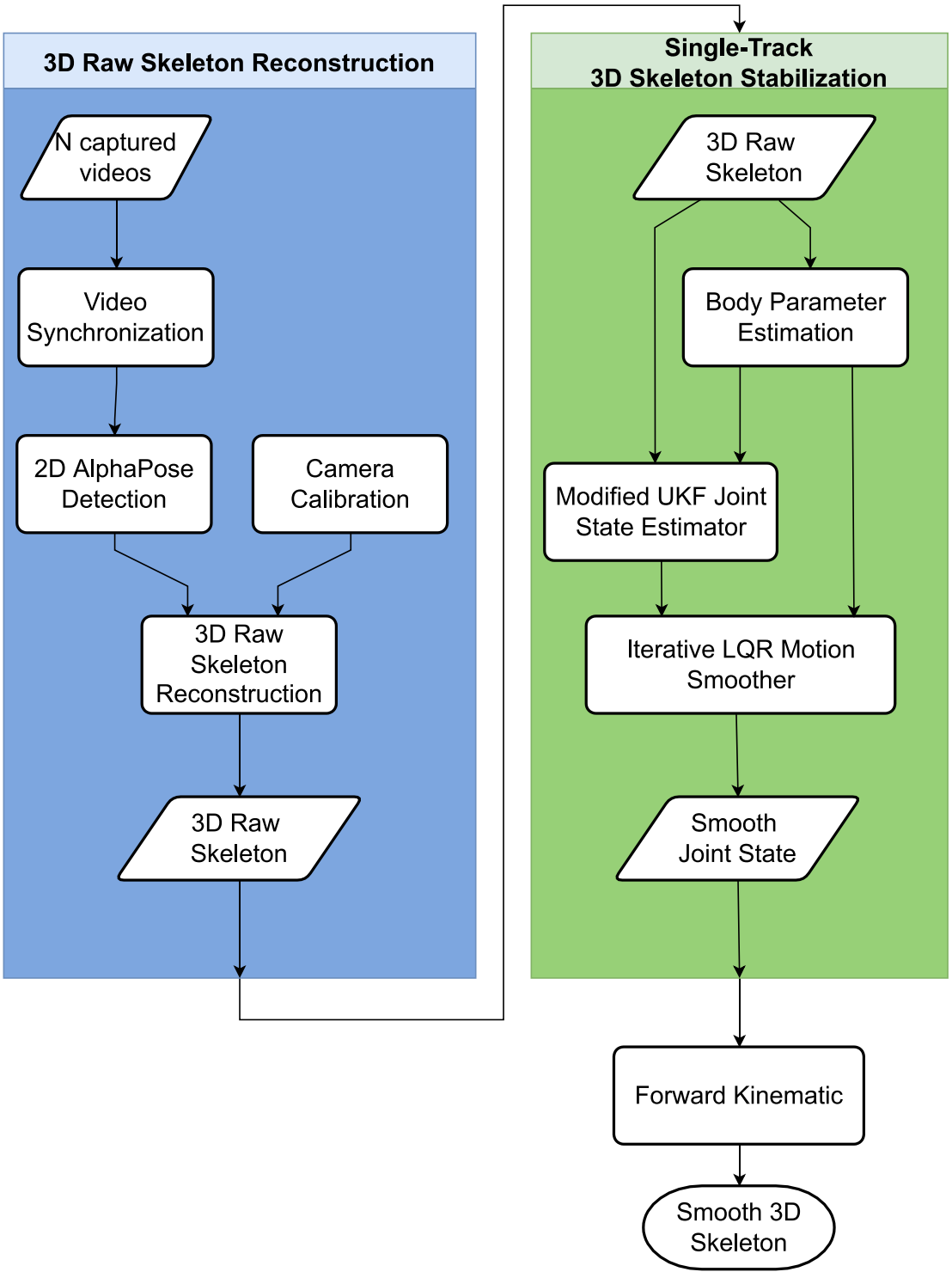


Figure 1.4: Overview of the proposed system.

doi:10.6342/NTU202202468

doi:10.6342/NTU202202468

1.4 Organization of the Thesis

This rest of this thesis is organized as the following statements. Chapter 2 presents the background and literature survey of the thesis. Chapter 3 introduces the related algorithms of the proposed method. Chapter 4 shows how we build the multi-view system and reconstruct the raw skeleton from the system. Chapter 5 describes the proposed single-track 3D human motion estimation in detail. Chapter 6 demonstrates the simulation and experiment results and analyzes the effects of the proposed methods in advance. Chapter 7 summarizes this thesis and points out the future works of this research.

doi:10.6342/NTU202202468

Chapter 2
Background and Literature Survey

In this chapter, the background and literature survey for this thesis is discussed. There are three topics in this chapter. Section 2.1 provides the state-of-the-art motion capture systems for sports. Section 2.2 presents how the other works estimate the human poses with vision information.

2.1 Motion Capture Systems

Motion capture systems are the systems that capture the 3D motion of the targets, for humans mainly. Depending on the applied scenarios, various motion capture systems have been developed. According to the measurement techniques, those systems can be divided into four categories: electromagnetic measurement systems (EMS), image processing systems (IMS), optoelectronic measurement systems (OMS), and inertial measurement units (IMU) [1: van der Kruk & Reijne 2018], as shown in Figure 2.1.

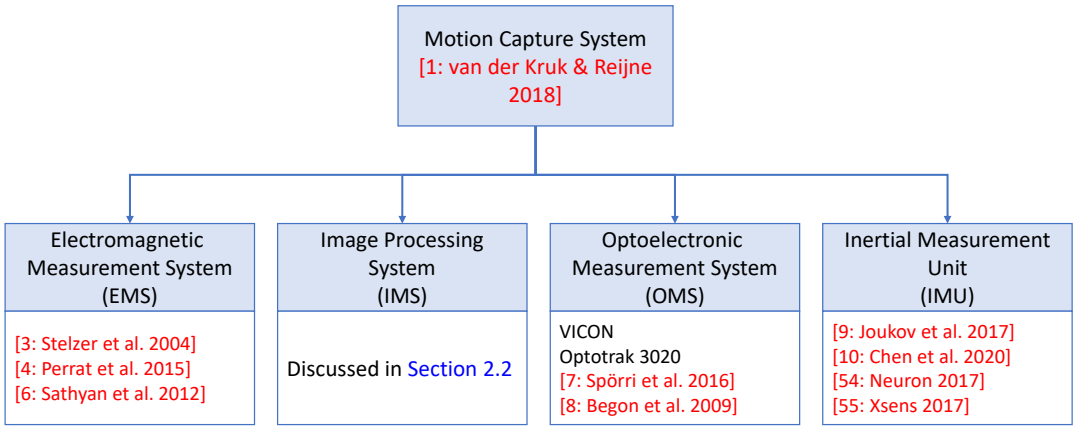


Figure 2.1: Classification of motion capture systems.

doi:10.6342/NTU202202468

2.1.1 Electromagnetic Measurement System (EMS)

The EMSs measure the positions with the time-of-flight of the electromagnetic waves – radio waves – traveling from the transponder to the base stations [3: Stelzer et al. 2004][4: Perrat et al. 2015]. Since the human body is transparent at the wave frequency [5: Schepers & Veltink 2010], this measurement technique suits crowded environments, such as team sports [6: Sathyan et al. 2012]. However, the accuracy of EMSs is relatively low among the four categories. The transponders needed to be installed on the moving targets is also another issue for some scenarios, like some official sport games.

2.1.2 Image Processing System (IMS)

The IMSs are also called vision-based methods. Rather than sensing the additional objects (markers or transponders) on targets, these systems sense the target, usually humans, with images or videos by computer vision algorithms directly. Therefore, they can fulfill markerless motion capture, which makes it have wider applied scenarios. The IMSs generally have better accuracy than the EMSs but worse than the OMSs. Computer vision algorithm development is a difficulty for IMSs. But with the growth of machine learning techniques, the IMSs become more and more popular in recent years, which would be described in detail in Section 2.2.

doi:10.6342/NTU202202468

2.1.3 Optoelectronic Measurement System (OMS)

The OMSs are the most accurate among those four categories and are often regarded as the gold standard in the literature. Those systems use infrared light to locate the markers on the moving targets with triangulation by multiple cameras. Dependent on whether the markers contain the source of light or not, the marker systems can be classified into passive, e.g. VICON, or active, e.g. Optotrak 3020. Without the additional cable and battery, the former has fewer limited motions. On the other hand, the latter performs more robust results than the passive ones.

To fulfill the high precision, most OMSs utilize several mounted cameras in a fixed frame. Consequentially, the accuracy of the systems is highly dependent on the experimental setup, such as distances between cameras and markers, calibration between cameras, positions and number of markers, and brightness of environment infrared. Owing to the sensitivity, the implementation spaces of OMSs are often limited in indoor environments, and their range is highly relative to the number of mounted cameras [7: Spörri et al. 2016]. To expand the applied range with fewer cameras, [8: Begon et al. 2009] used several mounted cameras on a movable frame and made the frame track the target with several markers fixed on the ground as the only global information.

doi:10.6342/NTU202202468

2.1.4 Inertial Measurement Unit (IMU)

Instead of installing the sensors away from the targets, the IMUs sense the targets' motions directly mounted on the targets. Thus, the implementation range would not be constrained by the equipment. However, the IMUs can't output the positions directly. Besides, the performance is dependent on the fusion filters. [79: Neuron 2022] and [80: Xsens 2022] utilize rigid-body models to estimate the positions of human bodies. [9: Joukov et al. 2017] uses Lie-group extend Kalman filter to solve the gimbal lock issue occurring for sphere joints. [10: Chen et al. 2020] developed a self-aligned algorithm for IMUs and solve the drifting problem for the IMUs.

2.2 3D Human Pose Estimation (HPE) with Vision

In recent years, human pose estimation has become more and more popular [11: Sarafianos et al. 2016]. The main reason causes from the increasing of new applications [12: Wang et al. 2021], such as human-robot interaction [13: Zhang 2012], autonomous driving [14: Kim et al. 2019][15: Du et al. 2019], sport performance analysis [16: Hwang et al. 2017][17: Rematas et al. 2018], etc. The vision-based human pose estimation has the advantage for wide application scenarios. Since these methods don't need any devices attached to human bodies, they are compatible with our daily usage.

In this field, the algorithms can be divided into two periods. From 2008 to 2015, most works developed several classical methods and few deep-learning-based methods

doi:10.6342/NTU202202468

[11: Sarafianos et al. 2016]. After 2016, the deep-learning-based methods for 3D human pose estimation progressed rapidly [12: Wang et al. 2021]. In this section, the works of literature would be introduced in the two periods respectively.

2.2.1 3D Human Pose Estimation in 2008-2015

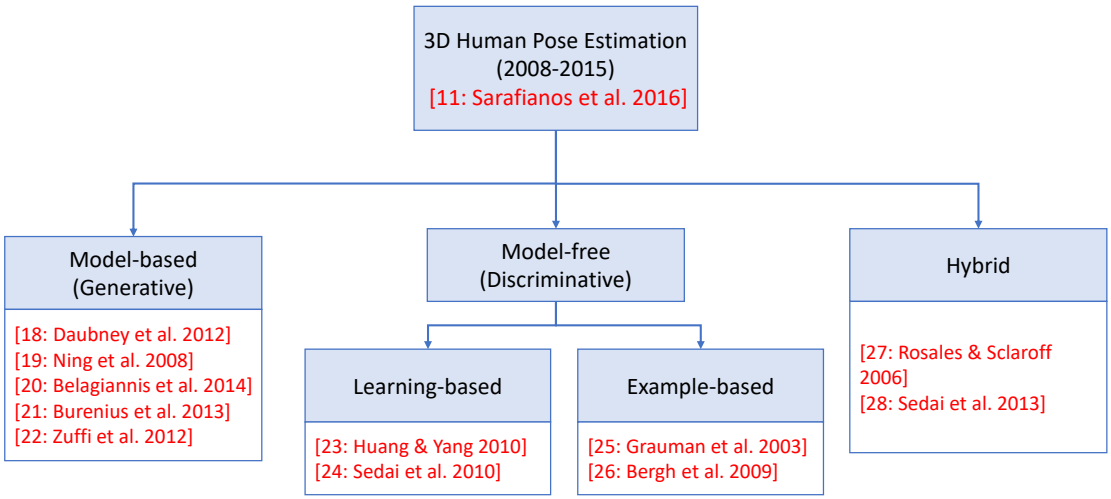


Figure 2.2: Classification of 3D human pose estimation methods from 2008 to 2015.

According to the algorithm, the vision-based methods for human pose estimation can be categorized as model-based, model-free and hybrid methods, as shown in Figure 2.2.

For the **model-based methods**, they are also referred as generative model approach in [11: Sarafianos et al. 2016]. These methods employ a known model based on prior information such as specific motion [18: Daubney et al. 2012] and context [19: Ning et al. 2008]. Another category of model-based methods is called part-based or bottom-up methods. Those methods regard the representation of human skeletons as a collection of body parts [20: Belagiannis et al. 2014][21: Burenius et al. 2013][22: Zuffi et al. 2012].

doi:10.6342/NTU202202468

For **model-free methods**, also known as discriminative approaches, they can be divided into learning-based and example-based. The learning-based methods learn the mapping function from the input images to the estimated 3D poses [23: Huang & Yang 2010][24: Sedai et al. 2010]. The example-based methods output the interpolation of candidates as the estimated poses [25: Grauman et al. 2003][26: Bergh et al. 2009], which can increase the robustness and speed for the estimation.

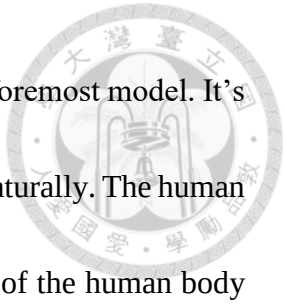
Finally, for the **hybrid methods**, they combine the generative and discriminative approaches to predict the pose more accurately. As the combination of two approaches, the observation of discriminative approaches would be verified with the generative approaches. [27: Rosales & Sclaroff 2006] and [28: Sedai et al. 2013] are examples of hybrid approaches. Besides, our proposed framework in this thesis can also be regarded as a hybrid approach.

2.2.2 3D Human Pose Estimation after 2016

With the growth of machine learning techniques and the large 3D human pose datasets, such as Human3.6M [29: Ionescu et al. 2014], CMU Panoptic [31: Joo et al. 2018], … etc., deep-learning-based methods become the mainstream for 3D human motion estimation in last few years.

There are three common human body models for those deep 3D human pose estimation approaches: skeleton-based model, SMPL (skinned multi-person linear) model,

doi:10.6342/NTU202202468



and surface-based model. The **skeleton-based model** is the first and foremost model. It's commonly used in 2D human motion estimation and extended to 3D naturally. The human skeleton is represented as a tree where the vertices are the keypoints of the human body and the edges are formed by connecting the adjacent joints. The keypoint definitions are different in different datasets. For the **SMPL model** [32: Loper et al. 2015], it is a triangulated mesh with 6890 vertices to represent the human skin. The mesh of SMPL models is parameterized by the shape and pose parameters. The 3D pose positions can be estimated by learning those parameters. For the last one, **surface-based model**, the most famous one is DensePose [33: Güler et al. 2018]. Considering the fact that the sparse correspondence of the image and keypoints may not be enough to capture the status of the human body for some applications, this model established the correspondences between the 3D positions of the human surfaces and the image pixels.

Based on the input data are a single frame or a sequence, monocular or multi-views, there are four types of 3D human pose estimation. The classification is illustrated in Figure 2.3.

doi:10.6342/NTU202202468

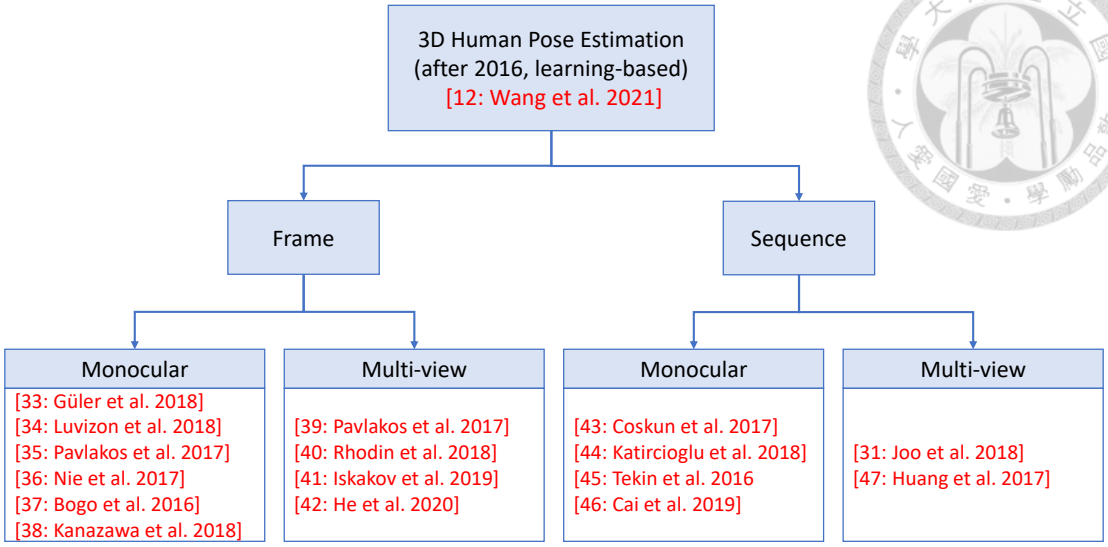


Figure 2.3: Classification of deep-learning-based 3D human pose estimation methods after 2016.

For the **single-frame monocular** type, the fact that a 2D pose may correspond to several different 3D poses is often regarded as an ill-defined problem. But due to the appealing low requirement of the image, there are still several methods have been developed. [34: Luvizon et al. 2018] and [35: Pavlakos et al. 2017] estimate the 3D human poses through an end-to-end network directly. [33: Güler et al. 2018] and [36: Nie et al. 2017] lift the 2D keypoints to form the corresponding 3D poses. [37: Bogo et al. 2016] and [38: Kanazawa et al. 2018] use the SMPL model to match the SMPL parameters and the images so that the keypoints on SMPL models can be projected to the corresponding 2D positions.

With the multi-view images, the **single-frame multi-view** type 3D HPE can reduce the ambiguity of depth significantly. On the other hand, the challenge for this type is how to fuse the information from multiple views. [39: Pavlakos et al. 2017] fused the multi-

doi:10.6342/NTU202202468

view 2D heatmaps by back-projecting to a common discretized 3D space with 3D pictorial structures models. The multi-view consistency is used in [40: Rhodin et al. 2018] as weak supervision, by forcing the network to predict the same pose from all views during training. Triangulation is one of the fundamental and popular techniques for 3D HPE. Some state-of-the-art methods in Human3.6M used this technique, such as [41: Iskakov et al. 2019] and [42: He et al. 2020].

For the **sequential monocular** 3D HPE, the inherent depth ambiguity also causes ill-defined problems like single-frame monocular 3D HPE. To reduce the ambiguity, temporal information, like invariant body shapes and motion continuity, is often used. To utilize them on artificial neural networks, various model architectures are implemented, such as long short-term memory (LSTM) [43: Coskun et al. 2017][44: Katircioglu et al. 2018], convolutional neural networks (CNN) [45: Tekin et al. 2016], and graph convolutional networks (GCN) [46: Cai et al. 2019]. [43: Coskun et al. 2017] proposed LSTM-KF and tried to learn the motion and noise model of Kalman filter from LSTM. [44: Katircioglu et al. 2018] used LSTM to impose the temporal constraint on the early features. [45: Tekin et al. 2016] concatenated the input sequences to form the spatial-temporal volume and extract the feature from the volume. With the skeleton model, [46: Cai et al. 2019] formed the spatial graph of a human skeleton and further connected the

doi:10.6342/NTU202202468

keypoints along the time axis to build a spatial-temporal graph. With the GCNs, the estimated poses are refined with spatial and temporal information.

For the **sequential multi-view** 3D HPE, it has better performance than single-view methods with the richness of the information. [31: Joo et al. 2018] used 480 synchronized VGA views to calculate the 3D position likelihood with the projection of the center voxel to all 2D views. [47: Huang et al. 2017] proposed SMPLify which fits SMPL model for all views independently and regularizes the motion in time.

Nonetheless, although the deep-learning-based methods have become the mainstream methods in the last few years. The in-the-wild scenarios are always a bottleneck for learning-based methods inherently. Not like 2D annotations, it's hard to construct large datasets for 3D annotations without marker-based vision systems. As mentioned in Section 2.1.3, those optoelectronic measurement systems are sensitive to the experiment setup. Therefore, a large accurate outdoor 3D dataset for human pose estimation is almost impossible. To overcome this problem, we proposed a non-learning 3D lifting and refining framework with the real-world well-perform 2D pose estimator, AlphaPose [48: Fang et al. 2017], to get rid of 3D annotation data.

doi:10.6342/NTU202202468

Chapter 3
Related Algorithms

In this chapter, the related algorithms used in this thesis would be introduced. To reconstruct the 3D raw skeleton from multi-view images, AlphaPose and epipolar geometry are presented in Section 3.1 and Section 3.2 respectively. The sphere fitting useful to estimate the unknown joint position is described in Section 3.3. Applied to the outlier component rejecting mechanism, the simple linear regression is mentioned in Section 3.5. To solve the state initialization with inversed kinematics, the 3D transformation estimation and rotation matrix decomposition are shown in Section 3.6 and Section 3.7. Finally, the unscented Kalman filter (UKF) and linear quadratic regulator are the main algorithms helping to filter the rough raw skeletons into smooth post skeletons and would be explained in Section 3.4 and Section 3.8.

doi:10.6342/NTU202202468

3.1 AlphaPose

AlphaPose [48: Fang et al. 2017] is the state-of-the-art algorithm for 2D multi-person pose estimation. As a two-step top-down method, the framework of AlphaPose combined YOLOv3 [49: Redmon & Farhadi 2018] as the region of interest (ROI) proposer for human detection and the symmetric spatial transformer network (SSTN) + single-person pose estimator (SPPE) to propose the corresponding poses of humans.

Using SPPE to estimate the human poses directly is a straightforward idea. However, the performances of the SPPE are sensitive to the bounding box of the proposed ROI. Since this issue, [48: Fang et al. 2017] proposed the spatial transformer network (STN) to adjust the proposed ROI from the input ROI image. The SPPE performs better with the adjusted ROI. To find estimated poses in the original ROI coordinates, the authors used the spatial de-transformer network (SDTN) to recover the original pose coordinates with the parameters in STN. With the STN and SDTN, the symmetric STN (SSTN) is formed.

Besides, to increase the robustness of AlphaPose for wild images, the authors introduce the pose-guided proposals generator (PGPG) to augment the imperfect human ROI proposals and set a fixed parallel SPPE module to penalize distance errors between the center of the ground truth bounding boxes and the estimated poses after STN during training. This process can make the STN try to propose the ROI around the human centers. Finally, because the human ROI proposer may generate highly overlapped ROIs for the

doi:10.6342/NTU202202468

same person, the authors proposed a parametric pose non-maximum-suppression (p-Pose NMS) to remove the lower-confidence poses overlapped with the other higher-confidence poses. The overall pipeline of AlphaPose is shown in Figure 3.1.

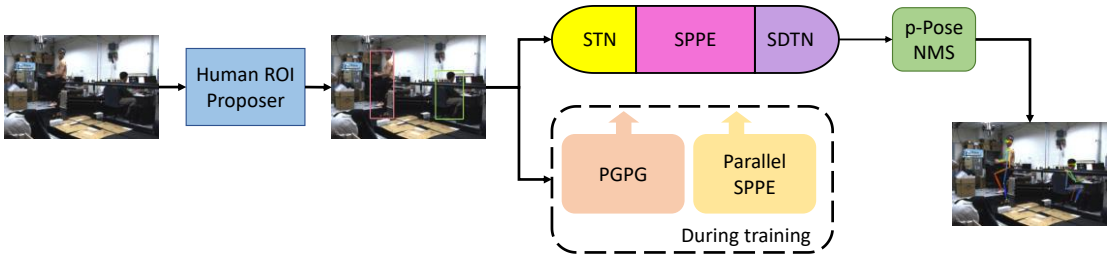


Figure 3.1: The Pipeline of AlphaPose

3.2 Epipolar Geometry

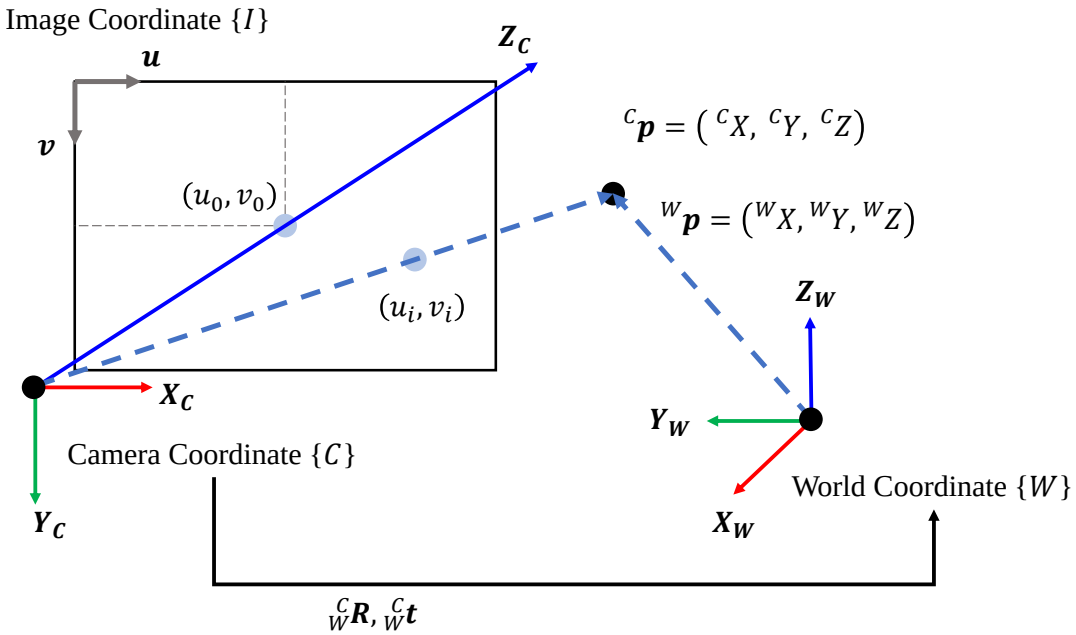


Figure 3.2: The Pinhole Camera Model

Refer to chapter 6 in [71: Hartley & Zisserman 2004], the projection from a point in the world to an image point can be formulated with Equation (3.1), where $\boldsymbol{K} \in \mathbb{R}^{3\times3}$ is

doi:10.6342/NTU202202468

the intrinsic parameters including the focal lengths f_x, f_y and lens center u_0, v_0 in the image coordinate, and the extrinsic parameters ${}^{c}_{w}\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and ${}^{c}_{w}\boldsymbol{t} \in \mathbb{R}^{3}$ represent the rotation and translation transformation from the world coordinate to camera coordinate. $[u \quad v]^T$ is the 2D image position of the projected point in the image coordinate. $[{}^{w}X \quad {}^{w}Y \quad {}^{w}Z]^T$ and $[{}^{c}X \quad {}^{c}Y \quad {}^{c}Z]^T$ represent the 3D point position in world coordinate and camera coordinate respectively.

$$
\begin{aligned}
{}^{c}Z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{21} & r_{31} & t_x \\ r_{12} & r_{22} & r_{32} & t_y \\ r_{13} & r_{23} & r_{33} & t_z \end{bmatrix}\begin{bmatrix} {}^{w}X \\ {}^{w}Y \\ {}^{w}Z \\ 1 \end{bmatrix} \\
&= \boldsymbol{K}\cdot[{}^{c}_{w}\boldsymbol{R}|{}^{c}_{w}\boldsymbol{t}]\begin{bmatrix} {}^{w}X \\ {}^{w}Y \\ {}^{w}Z \\ 1 \end{bmatrix} \\
&= \boldsymbol{K}\cdot\begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \end{bmatrix}
\end{aligned}
\tag{3.1}
$$

With the pinhole camera model in Figure 3.2, we can realize that the 3D position of a point projected on a camera may come from any points in the corresponding ray. Therefore, it's not possible to reconstruct the 3D points with only one camera.

To find the 3D positions, a multi-camera system is needed. The technique to reconstruct the 3D points with two cameras is called "epipolar geometry," shown in Figure 3.3.

doi:10.6342/NTU202202468

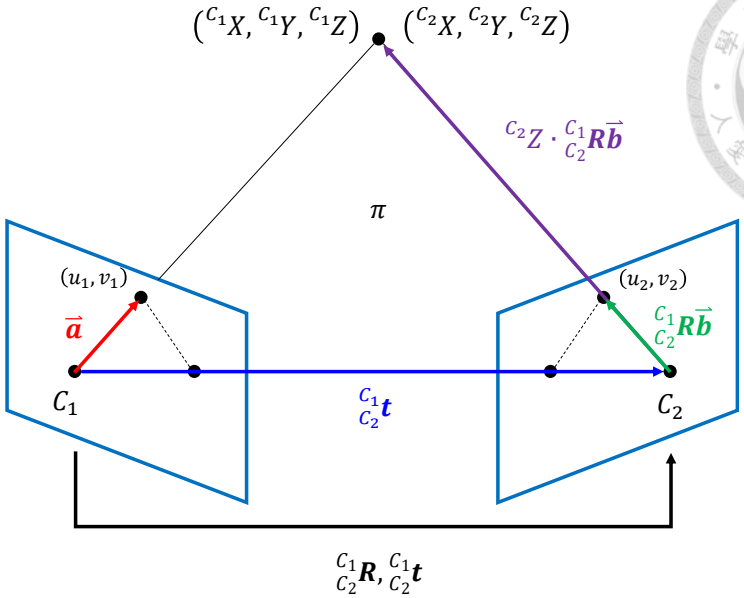


Figure 3.3: The Schematic Diagram of Epipolar Geometry

Since a camera can generate a ray as the candidates of a 3D point position, with two cameras whose rays are not parallel to each other, the 3D position of the point can be estimated at the intersection or the middle point of the common perpendicular of the two rays.

To formulate it, assuming the camera sensing is ideal, Equation (3.1) is re-written as Equation (3.2) and Equation (3.3) for camera C_1 and camera C_2:

$$\begin{bmatrix} {}^{C_1}X \\ {}^{C_1}Y \\ {}^{C_1}Z \end{bmatrix} = \boldsymbol{K_1}^{-1} \cdot {}^{C_1}Z \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = {}^{C_1}Z \cdot \vec{\boldsymbol{a}} \tag{3.2}$$

$$\begin{bmatrix} {}^{C_2}X \\ {}^{C_2}Y \\ {}^{C_2}Z \end{bmatrix} = \boldsymbol{K_2}^{-1} \cdot {}^{C_2}Z \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = {}^{C_2}Z \cdot \vec{\boldsymbol{b}} \tag{3.3}$$

With the known rigid body transformation $\left[{}^{C_1}_{C_2}\boldsymbol{R} | {}^{C_1}_{C_2}\boldsymbol{t}\right]$ from C_2 to C_1, Equation (3.4) is formed:

doi:10.6342/NTU202202468

$$
\begin{aligned}
{}^{C_1}Z \cdot \vec{\boldsymbol{a}} &= \begin{bmatrix} {}^{C_1}X \\ {}^{C_1}Y \\ {}^{C_1}Z \end{bmatrix} \\
&= {}^{C_1}_{C_2}\boldsymbol{R} \begin{bmatrix} {}^{C_2}X \\ {}^{C_2}Y \\ {}^{C_2}Z \end{bmatrix} + {}^{C_1}_{C_2}\boldsymbol{t} \qquad (3.4) \\
&= {}^{C_1}_{C_2}\boldsymbol{R}\left({}^{C_2}Z \cdot \vec{\boldsymbol{b}}\right) + {}^{C_1}_{C_2}\boldsymbol{t} \\
&= {}^{C_2}Z\,{}^{C_1}_{C_2}\boldsymbol{R}\vec{\boldsymbol{b}} + {}^{C_1}_{C_2}\boldsymbol{t}
\end{aligned}
$$

Since ${}^{C_1}Z$ is a scalar, we can find that $\vec{\boldsymbol{a}}$ is parallel to ${}^{C_2}Z \cdot {}^{C_1}_{C_2}\boldsymbol{R}\vec{\boldsymbol{b}} + {}^{C_1}_{C_2}\boldsymbol{t}$, which forms Equation (3.5):

$$
\vec{\boldsymbol{a}} \times \left({}^{C_2}Z \cdot {}^{C_1}_{C_2}\boldsymbol{R}\vec{\boldsymbol{b}} + {}^{C_1}_{C_2}\boldsymbol{t}\right) = \boldsymbol{0} \qquad (3.5)
$$

Therefore, the depth from C_2 can be calculated as the Equation (3.6):

$$
{}^{C_2}Z = \frac{\left\|\vec{\boldsymbol{a}} \times {}^{C_1}_{C_2}\boldsymbol{t}\right\|}{\left\|\vec{\boldsymbol{a}} \times {}^{C_1}_{C_2}\boldsymbol{R}\vec{\boldsymbol{b}}\right\|} \qquad (3.6)
$$

${}^{C_2}Z \geq 0$ since the projected point must be in the front of camera. Then, the 3D point position in C_1 and C_2 can be calculated by Equation (3.4) and Equation (3.3) with the calculated ${}^{C_2}Z$.

doi:10.6342/NTU202202468

3.3 Sphere fitting

Referring to [85: Jekel 2022], a sphere can be formulated as:

$$r^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 \tag{3.7}$$

where $[x_0 \quad y_0 \quad z_0]^T$ is the sphere center and r is the radius of the sphere.

By arranging the terms in Equation (3.7):

$$r^2 = x^2 - 2x_0x + x_0^2 + y^2 - 2y_0y + y_0^2 + z^2 - 2z_0z + z_0^2 \tag{3.8}$$

$$2x_0x + 2y_0y + 2z_0z + r^2 - x_0^2 - y_0^2 - z_0^2 = x^2 + y^2 + z^2 \tag{3.9}$$

$$[2x \quad 2y \quad 2z \quad 1]\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ r^2 - x_0^2 - y_0^2 - z_0^2 \end{bmatrix} = x^2 + y^2 + z^2 \tag{3.10}$$

Every point on the sphere should follow the relation in Equation (3.10). Therefore, when there are n points sampled on a sphere:

$$\begin{bmatrix} 2x_1 & 2y_1 & 2z_1 & 1 \\ 2x_2 & 2y_2 & 2z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 2x_n & 2y_n & 2z_n & 1 \end{bmatrix}\begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ r^2 - x_0^2 - y_0^2 - z_0^2 \end{bmatrix} = \begin{bmatrix} {x_1}^2 + {y_1}^2 + {z_1}^2 \\ {x_2}^2 + {y_2}^2 + {z_2}^2 \\ \vdots \\ {x_n}^2 + {y_n}^2 + {z_n}^2 \end{bmatrix} \tag{3.11}$$

Let

$$\boldsymbol{A} = \begin{bmatrix} 2x_1 & 2y_1 & 2z_1 & 1 \\ 2x_2 & 2y_2 & 2z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 2x_n & 2y_n & 2z_n & 1 \end{bmatrix}$$

$$\vec{\boldsymbol{s}} = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \\ r^2 - x_0^2 - y_0^2 - z_0^2 \end{bmatrix} \tag{3.12}$$

doi:10.6342/NTU202202468

$$\vec{\boldsymbol{f}} = \begin{bmatrix} {x_1}^2 + {y_1}^2 + {z_1}^2 \\ {x_2}^2 + {y_2}^2 + {z_2}^2 \\ \vdots \\ {x_n}^2 + {y_n}^2 + {z_n}^2 \end{bmatrix}$$

$$\boldsymbol{A}\vec{\boldsymbol{s}} = \vec{\boldsymbol{f}} \tag{3.13}$$

$$\vec{\boldsymbol{s}} = pinv(\boldsymbol{A}) \cdot \vec{\boldsymbol{f}} \tag{3.14}$$

where $pinv(\cdot)$ is the pseudo inverse function.

Therefore, the center $\boldsymbol{c} \in \mathbb{R}^3$ and the radius $r \in \mathbb{R}$ of the sphere can be estimated with $\vec{\boldsymbol{s}} \in \mathbb{R}^4$:

$$\begin{bmatrix} \boldsymbol{c} \\ r^2 - \|\boldsymbol{c}\|^2 \end{bmatrix} = \vec{\boldsymbol{s}} \tag{3.15}$$

doi:10.6342/NTU202202468

3.4 Unscented Kalman Filter

Unscented Kalman filter (UKF) [50: Julier & Uhlmann 2004] is a variation of Kalman filter for the nonlinear systems in Equation (3.16) and Equation (3.17).

$$\begin{cases} \boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{w}_t \\ \boldsymbol{z}_t = h(\boldsymbol{x}_t) + \boldsymbol{v}_t \end{cases} \tag{3.16}$$

$$\begin{cases} \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}) \\ \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}) \end{cases} \tag{3.17}$$

$\boldsymbol{x}_t \in \mathbb{R}^n$ and $\boldsymbol{z}_t \in \mathbb{R}^m$ are the system states and observations at the time t. $\boldsymbol{w}_t \in \mathbb{R}^n$ is the process noise sampled from a zero-mean Gaussian distribution with the nonnegative definite covariance $\boldsymbol{Q}$ at the time t. $\boldsymbol{v}_t \in \mathbb{R}^m$ is the measurement noise sampled from a zero-mean Gaussian distribution with the positive definite covariance $\boldsymbol{R}$ at the time t. $\boldsymbol{u}_t$ is the system input at the time t. $f(\cdot)$ is the state transition function, and $h(\cdot)$ is the observation function. Both $f(\cdot)$ and $h(\cdot)$ can be nonlinear.

In the UKF, the processes are composed of three steps: 1) initialization, 2) time update, and 3) measurement update.

1). **Initialization**:

During the time update, the previous estimated state $\hat{\boldsymbol{x}}_{t-1}$ and state covariance $\boldsymbol{P}_{t-1|t-1}$ are needed. However, the $\hat{\boldsymbol{x}}_0$ and $\boldsymbol{P}_{0|0}$ aren't estimated before the filtering process. Thus, they should be initialized as Equation (3.18) and Equation (3.19):

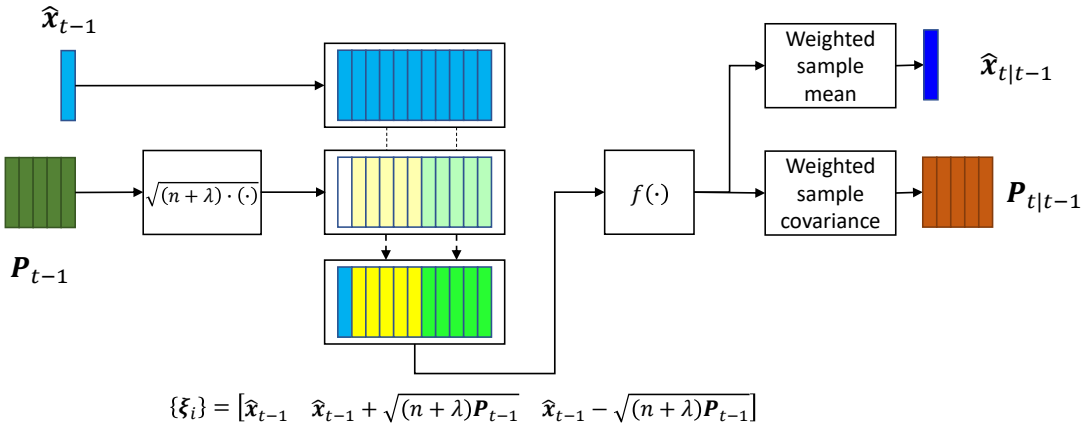
$$\hat{\boldsymbol{x}}_0 = E[\boldsymbol{x}_0] \tag{3.18}$$

doi:10.6342/NTU202202468

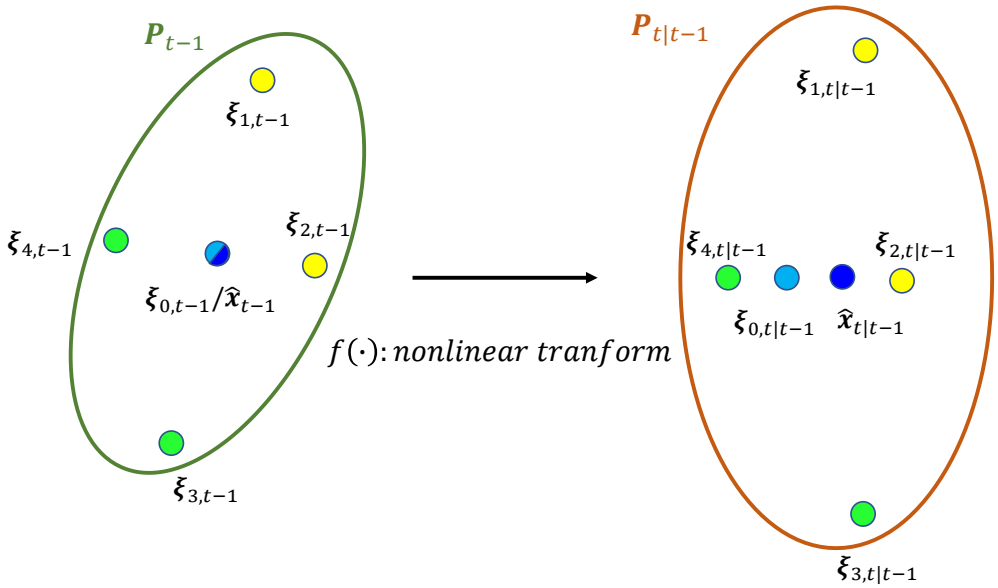
$$\boldsymbol{P}_0 = E[(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0)(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0)^T] \tag{3.19}$$

2). **Time update**:

Unlike the extended Kalman filter (EKF), the UKF uses unscented transform to linearize the nonlinear system without calculating the Jacobian of $f(\cdot)$ and $h(\cdot)$. An example of the unscented transform for state transition is illustrated in Figure 3.4.



(a)



(b)

Figure 3.4: The illustration of the unscented transform for state transition function (a) the block diagram (b) illustration for the relationship between sigma points and the predicted state and covariance

doi:10.6342/NTU202202468

Refer to [72: Wan et al. 2004], the first step of the unscented transform is to generate the sigma points $\boldsymbol{\xi}_{i,t-1}$ with Equation (3.20) while $\left(\sqrt{(\cdot)}\right)_i$ is the i −th column of the matrix square root calculated by Cholesky decomposition.

$$\begin{cases} \boldsymbol{\xi}_{0,t-1} = \hat{\boldsymbol{x}}_{t-1} \\ \boldsymbol{\xi}_{i,t-1} = \hat{\boldsymbol{x}}_{t-1} + \left(\sqrt{(n+\lambda)\boldsymbol{P}_{t-1|t-1}}\right)_i, & i = 1,2,\ldots,n \\ \boldsymbol{\xi}_{i,t-1} = \hat{\boldsymbol{x}}_{t-1} - \left(\sqrt{(n+\lambda)\boldsymbol{P}_{t-1|t-1}}\right)_i, & i = n+1, n+2,\ldots,2n \end{cases} \tag{3.20}$$

The second step is to do the state transition for every sigma point like Equation (3.21):

$$\boldsymbol{\xi}_{i,t|t-1} = f\left(\boldsymbol{\xi}_{i,t-1}, \boldsymbol{u}_t\right), \qquad i = 0,1,2,\ldots,2n \tag{3.21}$$

Then, the one-step predicted state $\hat{\boldsymbol{x}}_{t|t-1}$ and covariance $\boldsymbol{P}_{t|t-1}$ can be calculated by weighted average with ω_i^m and ω_i^c as Equation (3.22) and Equation (3.23).

$$\hat{\boldsymbol{x}}_{t|t-1} = \sum_{i=0}^{2n} \omega_i^m \boldsymbol{\xi}_{i,t|t-1} \tag{3.22}$$

$$\boldsymbol{P}_{t|t-1} = \left[\sum_{i=0}^{2n} \omega_i^c \left(\boldsymbol{\xi}_{i,t|t-1} - \hat{\boldsymbol{x}}_{t|t-1}\right)\left(\boldsymbol{\xi}_{i,t|t-1} - \hat{\boldsymbol{x}}_{t|t-1}\right)^T\right] + \boldsymbol{Q} \tag{3.23}$$

Next, the predicted observation $\hat{\boldsymbol{z}}_{t|t-1}$ and observation covariance $\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}$ can also be calculated as Equation (3.25) and Equation (3.26) from the unscented transform for observation function as Equation (3.24)

$$\boldsymbol{\gamma}_{i,t|t-1} = h\left(\boldsymbol{\xi}_{i,t|t-1}\right), \qquad i = 0,1,2,\ldots,2n \tag{3.24}$$

doi:10.6342/NTU202202468

$$\hat{\boldsymbol{z}}_{t|t-1} = \sum_{i=0}^{2n} \omega_i^m \boldsymbol{\gamma}_{i,t|t-1} \tag{3.25}$$

$$\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1} = \left[\sum_{i=0}^{2n} \omega_i^c \left(\boldsymbol{\gamma}_{i,t|t-1} - \hat{\boldsymbol{z}}_{t|t-1}\right)\left(\boldsymbol{\gamma}_{i,t|t-1} - \hat{\boldsymbol{z}}_{t|t-1}\right)^T\right] + \boldsymbol{R} \tag{3.26}$$

For the parameters occurring in Equation (3.20) - (3.26), they are defined as:

$$\begin{cases} \omega_0^m = \dfrac{\lambda}{n+\lambda} \\ \omega_0^c = \dfrac{\lambda}{n+\lambda} + (1-\alpha^2+\beta) \\ \omega_i^m = \omega_i^c = \dfrac{1}{2(n+\lambda)}, \quad i = 1,2,\ldots,2n \\ \lambda = \alpha^2(n+\kappa) - n \end{cases} \tag{3.27}$$

The parameters α, β, and κ are the handcrafted parameters. For Gaussian distribution, $\beta = 2$ is optimal. Therefore, in this thesis, these handcrafted parameters for UKF are set as $\alpha = 0.2$, $\beta = 2$, and $\kappa = 3 - n$.

3). **Measurement update**:

In the measurement update, the UKF will use the current observation $\boldsymbol{z}_t$ to correct the predicted estimation. However, the covariance between the state and observation $\boldsymbol{P}_{\hat{\boldsymbol{x}}\hat{\boldsymbol{z}},t|t-1}$ should be calculated first as:

$$\boldsymbol{P}_{\hat{\boldsymbol{x}}\hat{\boldsymbol{z}},t|t-1} = \sum_{i=0}^{2n} \omega_i^c \left(\boldsymbol{\xi}_{i,t|t-1} - \hat{\boldsymbol{x}}_{t|t-1}\right)\left(\boldsymbol{\gamma}_{i,t|t-1} - \hat{\boldsymbol{z}}_{t|t-1}\right)^T \in \mathbb{R}^{n\times m} \tag{3.28}$$

Then, the Kalman gain $\boldsymbol{K}_t$ would be calculated as:

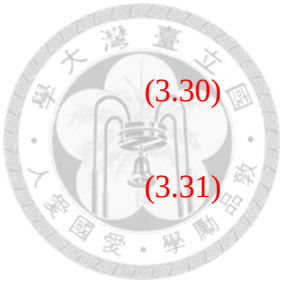
$$\boldsymbol{K}_t = \boldsymbol{P}_{\hat{\boldsymbol{x}}\hat{\boldsymbol{z}},t|t-1}\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}{}^{-1} \in \mathbb{R}^{n\times m} \tag{3.29}$$

Eventually, the corrected state $\hat{\boldsymbol{x}}_t$ and covariance $\boldsymbol{P}_t$ with the current observation $\boldsymbol{z}_t$ would be calculated as:

doi:10.6342/NTU202202468

$$\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\left(\boldsymbol{z}_t - \hat{\boldsymbol{z}}_{t|t-1}\right) \tag{3.30}$$

$$\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t \boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1} \boldsymbol{K}_t{}^T \tag{3.31}$$



3.5 Simple Linear Regression

Refer to [73: Chatterjee & Hadi 2006], for a set of points $\{(x_i, y_i) |\ i = 1, \dots, n\}$ where the means of x_i and y_i are μ_x and μ_y, we can find a line to fit the relationship between x and y among those points as Equation (3.32), where $p_{x,x}$ is the variance of x_i and $p_{y,x}$ is the covariance between x_i and y_i.

$$y = \hat{\alpha} + \hat{\beta} x \tag{3.32}$$

$$\begin{aligned} \hat{\beta} &= \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{n}(x_i - \mu_x)^2} \\ &= \frac{p_{y,x}}{p_{x,x}} \end{aligned} \tag{3.33}$$

$$\hat{\alpha} = \mu_y - \hat{\beta}\mu_x \tag{3.34}$$

Besides, we can calculate the correlation coefficient ρ_{xy} between x_i and y_i as:

$$\begin{aligned} \rho_{xy} &= \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_x)^2 \sum_{i=1}^{n}(y_i - \mu_y)^2}} \\ &= \frac{p_{y,x}}{\sqrt{p_{x,x} p_{y,y}}} \end{aligned} \tag{3.35}$$

Therefore, once we have the means μ_x, μ_y and the variances $p_{x,x}$, $p_{y,y}$ for x_i and y_i and their covariance $p_{y,x}$, every y_i can be estimated with x_i as:

$$\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i \tag{3.36}$$

doi:10.6342/NTU202202468

3.6 3D Transformation Matrix Estimation

Refer to [52: Gan & Dai 2011], a 3D transformation matrix ${}^{A}_{B}\boldsymbol{T}$ between two coordinates $\{A\}$ and $\{B\}$ can be estimated with $n \geq 4$ points $(\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_{n-1})$ whose positions are known in the both coordinates.

The 3D transformation matrix is composed of the rotation matrix ${}^{A}_{B}\boldsymbol{R} \in \mathbb{R}^{3\times3}$, the translation vector ${}^{A}_{B}\boldsymbol{t} \in \mathbb{R}^{3}$ and some constants as:

$$ {}^{A}_{B}\boldsymbol{T} = \begin{bmatrix} {}^{A}_{B}\boldsymbol{R} & {}^{A}_{B}\boldsymbol{t} \\ \boldsymbol{0}_{3\times1} & 1 \end{bmatrix} \tag{3.37} $$

For every point ${}^{B}\boldsymbol{p}_i$ in $\{B\}$, its position in $\{A\}$ can be written as:

$$ {}^{A}\boldsymbol{p}_i = {}^{A}_{B}\boldsymbol{R}\,{}^{B}\boldsymbol{p}_i + {}^{A}_{B}\boldsymbol{t} \quad \forall i = 0,1,2,\ldots,n-1 \tag{3.38} $$

The relative vectors from $\boldsymbol{p}_0$ to $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{n-1}$ can be written as:

$$ \begin{aligned} \left({}^{A}\boldsymbol{p}_i - {}^{A}\boldsymbol{p}_0\right) &= {}^{A}_{B}\boldsymbol{R}\left({}^{B}\boldsymbol{p}_i - {}^{B}\boldsymbol{p}_0\right) + \left({}^{A}_{B}\boldsymbol{t} - {}^{A}_{B}\boldsymbol{t}\right) \\ &= {}^{A}_{B}\boldsymbol{R}\left({}^{B}\boldsymbol{p}_i - {}^{B}\boldsymbol{p}_0\right) \\ &\forall i = 1,2,\ldots,n-1 \end{aligned} \tag{3.39} $$

Concatenating every relative vector, we can obtain the relative vector matrixes:

$$ \begin{cases} {}^{A}\boldsymbol{P} = \left[\left({}^{A}\boldsymbol{p}_1 - {}^{A}\boldsymbol{p}_0\right) \quad \left({}^{A}\boldsymbol{p}_2 - {}^{A}\boldsymbol{p}_0\right) \quad \cdots \quad \left({}^{A}\boldsymbol{p}_3 - {}^{A}\boldsymbol{p}_0\right)\right] \in \mathbb{R}^{3\times(n-1)} \\ {}^{B}\boldsymbol{P} = \left[\left({}^{B}\boldsymbol{p}_1 - {}^{B}\boldsymbol{p}_0\right) \quad \left({}^{B}\boldsymbol{p}_2 - {}^{B}\boldsymbol{p}_0\right) \quad \cdots \quad \left({}^{B}\boldsymbol{p}_3 - {}^{B}\boldsymbol{p}_0\right)\right] \in \mathbb{R}^{3\times(n-1)} \end{cases} \tag{3.40} $$

And their transformation equation is:

$$ {}^{A}\boldsymbol{P} = {}^{A}_{B}\boldsymbol{R}\,{}^{B}\boldsymbol{P} \tag{3.41} $$

To estimate the rotation matrix ${}^{A}_{B}\widehat{\boldsymbol{R}}$, it's equal:

$$ {}^{A}_{B}\widehat{\boldsymbol{R}} = {}^{A}\boldsymbol{P} \cdot pinv\left({}^{B}\boldsymbol{P}\right) \tag{3.42} $$

doi:10.6342/NTU202202468

$pinv(\cdot)$ is the pseudo inverse function for the matrixes.

Then, the translation vector ${}^{A}_{B}\hat{\boldsymbol{t}}$ can be estimated by:

$$ {}^{A}_{B}\hat{\boldsymbol{t}} = \frac{\mathbf{1}}{n}\sum_{i=0}^{n-1} {}^{A}\boldsymbol{p}_i - {}^{A}_{B}\widehat{\boldsymbol{R}}\,{}^{B}\boldsymbol{p}_i \tag{3.43} $$

Therefore, the transformation matrix ${}^{A}_{B}\boldsymbol{T}$ between $\{A\}$ and $\{B\}$ can be estimated as:

$$ {}^{A}_{B}\widehat{\boldsymbol{T}} = \begin{bmatrix} {}^{A}_{B}\widehat{\boldsymbol{R}} & {}^{A}_{B}\hat{\boldsymbol{t}} \\ \mathbf{0}_{3\times1} & 1 \end{bmatrix} \tag{3.44} $$

doi:10.6342/NTU202202468

3.7 Rotation Matrix Decomposition

Refer to [53: Eberly 2008], for a rotation matrix $\boldsymbol{R}$, it can be decomposed as three rotation matrixes along the repeated or different axis $\boldsymbol{R}_x(\theta)$, $\boldsymbol{R}_y(\theta)$, or $\boldsymbol{R}_z(\theta)$.

The rotation matrix along the x-axis with the angle θ is:

$$\boldsymbol{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sin\,\theta \\ 0 & sin\,\theta & cos\,\theta \end{bmatrix} \tag{3.45}$$

The rotation matrix along the y-axis with the angle θ is:

$$\boldsymbol{R}_y(\theta) = \begin{bmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{bmatrix} \tag{3.46}$$

The rotation matrix along the z-axis with the angle θ is:

$$\boldsymbol{R}_z(\theta) = \begin{bmatrix} cos\,\theta & -sin\,\theta & 0 \\ sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.47}$$

For roll-pitch-yaw representation, the rotation matrix $\boldsymbol{R}$ should be decomposed as:

$$\boldsymbol{R} = \boldsymbol{R}_z(\gamma)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha) \tag{3.48}$$

while α, β, γ are calculated with:

Algorithm 3.1: ZYX Rotation Decomposition
Input: rotation matrix $\boldsymbol{R}$ **Output**: the rotation angle α, β, γ while $\boldsymbol{R} = \boldsymbol{R}_z(\gamma)\boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha)$
Note: r_{ij} is the element of $\boldsymbol{R}$ in the i-th row and the j-th column

if $r_{31} < 1$ **do**
  if $r_{31} > -1$ **do**
  $\beta \leftarrow \arcsin(-r_{31})$
  $\gamma \leftarrow \text{arctan2}(r_{21}, r_{11})$
  $\alpha \leftarrow \text{arctan2}(r_{32}, r_{33})$
  else do

doi:10.6342/NTU202202468

7:	$\beta \leftarrow {}^{\pi}/_{2}$
8:	$\gamma \leftarrow -\arctan2(-r_{23}, r_{22})$
9:	$\alpha \leftarrow 0$
10:	**if end**
11:	**else do**
12:	$\beta \leftarrow -{}^{\pi}/_{2}$
13:	$\gamma \leftarrow \arctan2(-r_{23}, r_{22})$
14:	$\alpha \leftarrow 0$
15:	**if end**
16:	**return** $\alpha,\ \beta,\ \gamma$

For another representation used in this thesis $\boldsymbol{R} = \boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha)\boldsymbol{R}_z(\gamma)$, the decomposition algorithm is:

Algorithm 3.2: YXZ Rotation Decomposition

Input: rotation matrix $\boldsymbol{R}$

Output: the rotation angle $\alpha,\ \beta,\ \gamma$ while $\boldsymbol{R} = \boldsymbol{R}_y(\beta)\boldsymbol{R}_x(\alpha)\boldsymbol{R}_z(\gamma)$

Note: r_{ij} is the element of $\boldsymbol{R}$ in the i-th row and the j-th column

1:	**if** $r_{23} < 1$ **do**
2:	**if** $r_{23} > -1$ **do**
3:	$\alpha \leftarrow \arcsin(-r_{23})$
4:	$\beta \leftarrow \arctan2(r_{13}, r_{33})$
5:	$\gamma \leftarrow \arctan2(r_{21}, r_{22})$
6:	**else do**
7:	$\alpha \leftarrow {}^{\pi}/_{2}$
8:	$\beta \leftarrow -\arctan2(-r_{12}, r_{11})$
9:	$\gamma \leftarrow 0$
10:	**if end**
11:	**else do**
12:	$\alpha \leftarrow -{}^{\pi}/_{2}$
13:	$\beta \leftarrow \arctan2(-r_{12}, r_{11})$
14:	$\gamma \leftarrow 0$
15:	**if end**
16:	**return** $\alpha,\ \beta,\ \gamma$

doi:10.6342/NTU202202468

3.8 Linear Quadratic Regulator

Refer to [74: Anderson & Moore 2007], discrete linear quadratic regulator (LQR) is an optimization problem formulated in Equation (3.49):

$$\min_{\boldsymbol{u}_t} \frac{1}{2}\left[\sum_{t=0}^{N-1} \boldsymbol{x}_t{}^T \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{u}_t{}^T \boldsymbol{R}\boldsymbol{u}_t\right] + \frac{1}{2}\boldsymbol{x}_N{}^T \boldsymbol{Q}_N \boldsymbol{x}_N$$
$$\text{s.t. } \boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t,\ t = 0,1,\dots,N-1 \text{ given } \boldsymbol{x}_0 \tag{3.49}$$

By minimizing the quadratic form of the state $\boldsymbol{x}_t$ and control input $\boldsymbol{u}_t$ for $t = 0,1,\dots,N-1$ plus the terminal state cost $\boldsymbol{x}_N{}^T \boldsymbol{Q}_N \boldsymbol{x}_N$, LQR finds the optimal solution while the state dynamic is fit.

To solve this optimization problem, we can easily implement Lagrange multiplier:

$$L = \frac{1}{2}\left[\sum_{t=0}^{N-1} \boldsymbol{x}_t{}^T \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{u}_t{}^T \boldsymbol{R}\boldsymbol{u}_t\right] + \frac{1}{2}\boldsymbol{x}_N{}^T \boldsymbol{Q}_N \boldsymbol{x}_N + \left[\sum_{t=0}^{N-1} \boldsymbol{\lambda}_{t+1}{}^T (\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t - \boldsymbol{x}_{t+1})\right] \tag{3.50}$$

Thus, the constrained optimization problem becomes into an unconstrained optimization problem:

$$\min_{\boldsymbol{x}_t, \boldsymbol{x}_N, \boldsymbol{u}_t, \boldsymbol{\lambda}_{t+1}} L = \min_{\boldsymbol{x}_t, \boldsymbol{x}_N, \boldsymbol{u}_t, \boldsymbol{\lambda}_{t+1}} \frac{1}{2}\left[\sum_{t=0}^{N-1} \boldsymbol{x}_t{}^T \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{u}_t{}^T \boldsymbol{R}\boldsymbol{u}_t\right] + \frac{1}{2}\boldsymbol{x}_N{}^T \boldsymbol{Q}_N \boldsymbol{x}_N + \left[\sum_{t=0}^{N-1} \boldsymbol{\lambda}_{t+1}{}^T (\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t - \boldsymbol{x}_{t+1})\right] \tag{3.51}$$

For Equation (3.51), the minimal point of L can be found by taking the partial derivative for $\boldsymbol{x}_t$, $\boldsymbol{x}_N$, $\boldsymbol{u}_t$ and $\boldsymbol{\lambda}_t$ equal to $\boldsymbol{0}$.

doi:10.6342/NTU202202468

$$\frac{\partial L}{\partial \boldsymbol{x}_t} = \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{A}^T\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t = \boldsymbol{0}, \qquad t = 0,1,\dots,N-1 \tag{3.52}$$

$$\frac{\partial L}{\partial \boldsymbol{x}_N} = \boldsymbol{Q}_N\boldsymbol{x}_N - \boldsymbol{\lambda}_N = \boldsymbol{0} \tag{3.53}$$

$$\frac{\partial L}{\partial \boldsymbol{u}_t} = \boldsymbol{R}\boldsymbol{u}_t + \boldsymbol{B}^T\boldsymbol{\lambda}_{t+1} = \boldsymbol{0}, \qquad t = 0,1,\dots,N-1 \tag{3.54}$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_{t+1}} = \boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{B}\boldsymbol{u}_t - \boldsymbol{x}_{t+1} = \boldsymbol{0}, \qquad t = 0,1,\dots,N-1 \tag{3.55}$$

With Equation (3.54), the optimal control is:

$$\boldsymbol{u}_t = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1} \tag{3.56}$$

After arranging Equation (3.52) - (3.56) and the given $\boldsymbol{x}_0$, we can get the dynamic and the boundary conditions of the state $\boldsymbol{x}_t$ and the co-state $\boldsymbol{\lambda}_t$:

$$\begin{cases} \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{A}^T\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t, & \boldsymbol{\lambda}_N = \boldsymbol{Q}_N\boldsymbol{x}_N \\ \boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1} = \boldsymbol{x}_{t+1}, & \text{given } \boldsymbol{x}_0 \\ \forall t = 0,1,\dots,N-1 \end{cases} \tag{3.57}$$

Assume $\boldsymbol{\lambda}_t = \boldsymbol{P}_t\boldsymbol{x}_t,\ \boldsymbol{P}_t \in \mathbb{R}^{n\times n}$:

$$\boldsymbol{P}_N = \boldsymbol{Q}_N, \qquad for\ t = N \tag{3.58}$$

For $t = 0,1,\dots,N-1$

$$\begin{aligned} \boldsymbol{\lambda}_{t+1} &= \boldsymbol{P}_{t+1}\boldsymbol{x}_{t+1} \\ &= \boldsymbol{P}_{t+1}(\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1}) \\ &= \boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1} \end{aligned} \tag{3.59}$$

$$(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)\boldsymbol{\lambda}_{t+1} = \boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t \tag{3.60}$$

$$\boldsymbol{\lambda}_{t+1} = (\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t \tag{3.61}$$

With the co-state dynamic in Equation (3.57),

$$\begin{aligned} \boldsymbol{\lambda}_t &= \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{A}^T\boldsymbol{\lambda}_{t+1} \\ &= \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t \\ &= [\boldsymbol{Q} + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{P}_{t+1}\boldsymbol{A}]\boldsymbol{x}_t \\ &= \boldsymbol{P}_t\boldsymbol{x}_t \end{aligned} \tag{3.62}$$

doi:10.6342/NTU202202468

Therefore, $\boldsymbol{\lambda_t = P_t x_t}$ is proved where $\boldsymbol{P_t}$ is:

$$\boldsymbol{P_t} = \begin{cases} \boldsymbol{Q_N}, & t = N \\ \boldsymbol{Q + A^T(I + P_{t+1}BR^{-1}B^T)^{-1}P_{t+1}A}, & t = 0,1,\dots,N-1 \end{cases} \tag{3.63}$$

Then, the optimal state and co-state can be solved after solving $\boldsymbol{P_t}$ and the given $\boldsymbol{x_0}$:

$$\begin{aligned} \boldsymbol{x_{t+1}} &= \boldsymbol{Ax_t - BR^{-1}B^T\lambda_{t+1}} \\ &= \boldsymbol{Ax_t - BR^{-1}B^TP_{t+1}x_{t+1}} \end{aligned} \tag{3.64}$$

$$\boldsymbol{(I + BR^{-1}B^TP_{t+1})x_{t+1} = Ax_t} \tag{3.65}$$

$$\boldsymbol{x_{t+1} = (I + BR^{-1}B^TP_{t+1})^{-1}Ax_t} \tag{3.66}$$

$$\boldsymbol{\lambda_{t+1} = P_{t+1}x_{t+1}} \tag{3.67}$$

In addition, since the state $\boldsymbol{x_t}$ close to 0 is not always desirable, there is a variation of LQR for state tracking problem as Equation (3.42) where $\boldsymbol{\hat{x}_t}$ is the desirable state at time t.

$$\begin{aligned} &\min_{\boldsymbol{u_t}} \frac{1}{2}\left[\sum_{t=0}^{N-1}(\boldsymbol{x_t - \hat{x}_t})^T\boldsymbol{Q}(\boldsymbol{x_t - \hat{x}_t}) + \boldsymbol{u_t}^T\boldsymbol{Ru_t}\right] + \frac{1}{2}(\boldsymbol{x_N - \hat{x}_N})^T\boldsymbol{Q_N}(\boldsymbol{x_N - \hat{x}_N}) \\ &\text{s.t. } \boldsymbol{x_{t+1} = Ax_t + Bu_t},\ t = 0,1,\dots,N-1 \text{ given } \boldsymbol{x_0} \end{aligned} \tag{3.68}$$

Similar to the original LQR, we can also solve the state tracking LQR with Lagrange multiplier:

$$\begin{aligned} L = &\frac{1}{2}\left[\sum_{t=0}^{N-1}(\boldsymbol{x_t - \hat{x}_t})^T\boldsymbol{Q}(\boldsymbol{x_t - \hat{x}_t}) + \boldsymbol{u_t}^T\boldsymbol{Ru_t}\right] + \frac{1}{2}(\boldsymbol{x_N - \hat{x}_N})^T\boldsymbol{Q_N}(\boldsymbol{x_N - \hat{x}_N}) \\ &+ \left[\sum_{t=0}^{N-1}\boldsymbol{\lambda_{t+1}}^T(\boldsymbol{Ax_t + Bu_t - x_{t+1}})\right] \end{aligned} \tag{3.69}$$

doi:10.6342/NTU202202468

$$\min_{\boldsymbol{x}_t,\boldsymbol{x}_N,\boldsymbol{u}_t,\boldsymbol{\lambda}_{t+1}} L = \min_{\boldsymbol{x}_t,\boldsymbol{x}_N,\boldsymbol{u}_t,\boldsymbol{\lambda}_{t+1}} L \frac{1}{2}\left[\sum_{t=0}^{N-1}(\boldsymbol{x}_t-\hat{\boldsymbol{x}}_t)^T\boldsymbol{Q}(\boldsymbol{x}_t-\hat{\boldsymbol{x}}_t)+\boldsymbol{u}_t{}^T\boldsymbol{R}\boldsymbol{u}_t\right]$$
$$+\frac{1}{2}(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N)^T\boldsymbol{Q}_N(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N) \tag{3.70}$$
$$+\left[\sum_{t=0}^{N-1}\boldsymbol{\lambda}_{t+1}{}^T(\boldsymbol{A}\boldsymbol{x}_t+\boldsymbol{B}\boldsymbol{u}_t-\boldsymbol{x}_{t+1})\right]$$

Likewise, take the partial derivative equal to $\mathbf{0}$ for the optimal point:

$$\frac{\partial L}{\partial \boldsymbol{x}_t}=\boldsymbol{Q}(\boldsymbol{x}_t-\hat{\boldsymbol{x}}_t)+\boldsymbol{A}^T\boldsymbol{\lambda}_{t+1}-\boldsymbol{\lambda}_t=\mathbf{0},\qquad t=0,1,\dots,N-1 \tag{3.71}$$

$$\frac{\partial L}{\partial \boldsymbol{x}_N}=\boldsymbol{Q}_N(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N)-\boldsymbol{\lambda}_N=\mathbf{0} \tag{3.72}$$

$$\frac{\partial L}{\partial \boldsymbol{u}_t}=\boldsymbol{R}\boldsymbol{u}_t+\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1}=\mathbf{0},\qquad t=0,1,\dots,N-1 \tag{3.73}$$

$$\frac{\partial L}{\partial \boldsymbol{\lambda}_{t+1}}=\boldsymbol{A}\boldsymbol{x}_t+\boldsymbol{B}\boldsymbol{u}_t-\boldsymbol{x}_{t+1}=\mathbf{0},\qquad t=0,1,\dots,N-1 \tag{3.74}$$

With Equation (3.73), the optimal control is:

$$\boldsymbol{u}_t=-\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1} \tag{3.75}$$

After arranging Equation (3.71) - (3.75) and the given $\boldsymbol{x}_0$, we can get the dynamic and the boundary conditions of the state $\boldsymbol{x}_t$ and the co-state $\boldsymbol{\lambda}_t$:

$$\begin{cases}\boldsymbol{Q}(\boldsymbol{x}_t-\hat{\boldsymbol{x}}_t)+\boldsymbol{A}^T\boldsymbol{\lambda}_{t+1}=\boldsymbol{\lambda}_t, & \boldsymbol{\lambda}_N=\boldsymbol{Q}_N\boldsymbol{x}_N-\boldsymbol{Q}_N\hat{\boldsymbol{x}}_N\\ \boldsymbol{A}\boldsymbol{x}_t-\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1}=\boldsymbol{x}_{t+1}, & \text{given } \boldsymbol{x}_0\\ \forall t=0,1,\dots,N-1\end{cases} \tag{3.76}$$

In this problem, we assume $\boldsymbol{\lambda}_t=\boldsymbol{P}_t\boldsymbol{x}_t+\boldsymbol{c}_t$, $\boldsymbol{P}_t\in\mathbb{R}^{n\times n}$ and $\boldsymbol{c}_t\in\mathbb{R}^n$:

$$\begin{cases}\boldsymbol{P}_N=\boldsymbol{Q}_N\\ \boldsymbol{c}_t=-\boldsymbol{Q}_N\hat{\boldsymbol{x}}_N\end{cases},\qquad for\ t=N \tag{3.77}$$

For $t=0,1,\dots,N-1$

doi:10.6342/NTU202202468

$$\begin{aligned}\boldsymbol{\lambda}_{t+1} &= \boldsymbol{P}_{t+1}\boldsymbol{x}_{t+1} + \boldsymbol{c}_{t+1} \\ &= \boldsymbol{P}_{t+1}(\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1}) + \boldsymbol{c}_{t+1} \\ &= \boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{c}_{t+1} - \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1}\end{aligned} \tag{3.78}$$

$$(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)\boldsymbol{\lambda}_{t+1} = \boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t + \boldsymbol{c}_{t+1} \tag{3.79}$$

$$\boldsymbol{\lambda}_{t+1} = (\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t + (\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{c}_{t+1} \tag{3.80}$$

With the co-state dynamic in Equation (3.76),

$$\begin{aligned}\boldsymbol{\lambda}_t &= \boldsymbol{Q}(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t) + \boldsymbol{A}^T\boldsymbol{\lambda}_{t+1} \\ &= \boldsymbol{Q}\boldsymbol{x}_t + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{P}_{t+1}\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{Q}\hat{\boldsymbol{x}}_t \\ &\qquad + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{c}_{t+1} \\ &= [\boldsymbol{Q} + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{P}_{t+1}\boldsymbol{A}]\boldsymbol{x}_t - \boldsymbol{Q}\hat{\boldsymbol{x}}_t \\ &\qquad + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{c}_{t+1} \\ &= \boldsymbol{P}_t\boldsymbol{x}_t + \boldsymbol{c}_t\end{aligned} \tag{3.81}$$

Therefore, $\boldsymbol{\lambda}_t = \boldsymbol{P}_t\boldsymbol{x}_t + \boldsymbol{c}_t$ is proved where $\boldsymbol{P}_t$ and $\boldsymbol{c}_t$ are:

$$\boldsymbol{P}_t = \begin{cases} \boldsymbol{Q}_N, & t = N \\ \boldsymbol{Q} + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{P}_{t+1}\boldsymbol{A}, & t = 0,1,\dots,N-1 \end{cases} \tag{3.82}$$

$$\boldsymbol{c}_t = \begin{cases} -\boldsymbol{Q}_N\hat{\boldsymbol{x}}_N, & t = N \\ -\boldsymbol{Q}\hat{\boldsymbol{x}}_t + \boldsymbol{A}^T(\boldsymbol{I} + \boldsymbol{P}_{t+1}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T)^{-1}\boldsymbol{c}_{t+1}, & t = 0,1,\dots,N-1 \end{cases} \tag{3.83}$$

Then, the optimal tracking state be solved with $\boldsymbol{P}_t$, $\boldsymbol{c}_t$ and the given $\boldsymbol{x}_0$:

$$\begin{aligned}\boldsymbol{x}_{t+1} &= \boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{\lambda}_{t+1} \\ &= \boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T(\boldsymbol{P}_{t+1}\boldsymbol{x}_{t+1} + \boldsymbol{c}_{t+1})\end{aligned} \tag{3.84}$$

$$(\boldsymbol{I} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}_{t+1})\boldsymbol{x}_{t+1} = \boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{c}_{t+1} \tag{3.85}$$

$$\boldsymbol{x}_{t+1} = (\boldsymbol{I} + \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}_{t+1})^{-1}(\boldsymbol{A}\boldsymbol{x}_t - \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{c}_{t+1}) \tag{3.86}$$

doi:10.6342/NTU202202468

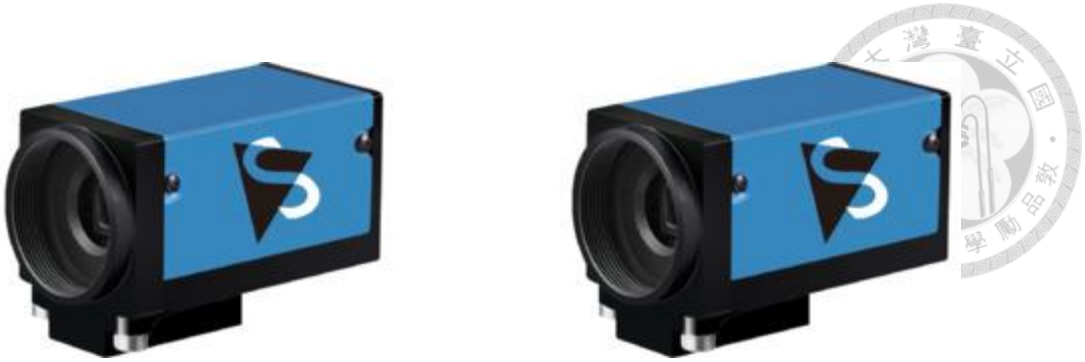
Chapter 4
3D Raw Skeleton Reconstruction

In this chapter, the proposed 3D human motion capture system and a rough 3D skeleton reconstruction method would be introduced. An overview of the multi-view system we build will be presented in Section 4.1. Next, in Section 4.2, an easy approach to calibrate and synchronize the system would be shown. Finally, to capture the 3D data with the proposed system, the 3D raw human pose reconstruction would be described in Section 4.3.

4.1 System structure of multi-view system

The cameras used in our multi-view system are the GigE industrial cameras as shown in Figure 4.1. The capture frequency can reach 300 Hz while the output image size is set as 720x540 pixels or 640x480 pixels. Besides the image capturing, these cameras can also record the capturing time for each frame with respect to the clock in the cameras. This function would increase the precision for synchronization and be introduced in Section 4.2.2.

doi:10.6342/NTU202202468



(a) (b)

Figure 4.1: The GigE industrial cameras used in this thesis from [81: The Imaging Source 2022]
(a) color camera DFK 33GX287
(b) monochrome camera DMK 33GX287

In addition, the lenses used in our system are listed in Table 4.1. Depending on the applied ranges, these lenses are selected to fit the best field of view (FOV) containing the whole activity range of the human. For our portable small-size capture system illustrated in Figure 4.4, we use the lenses MI-03524MP C to fulfill the small-range data recording. For the baseball field implementation, since the cameras should be installed out of the field, the other long-focal-length lenses listed in Table 4.1 are selected.

Table 4.1
Datasheet of the industrial lenses used in this thesis
from [82: Sure Technology Corporation 2022]

Lens Model	Focal Length (mm)	Nearest Working Distance (m)
MI-03524MP C	3.5	Not provided
MI-3514MP	35	0.3
MI-5018MP	50	0.5
MI-7528MP	75	1.1

doi:10.6342/NTU202202468

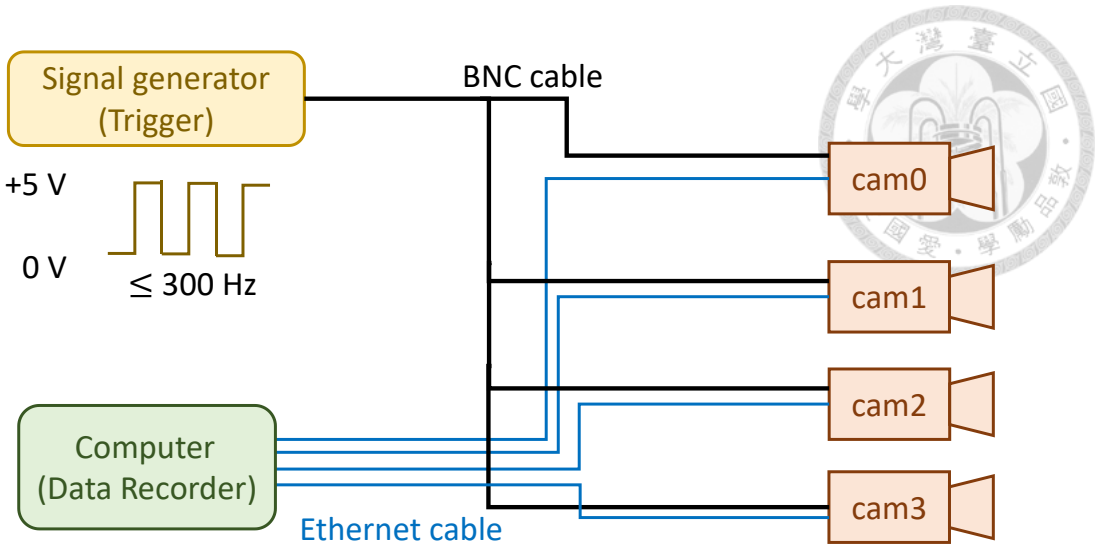
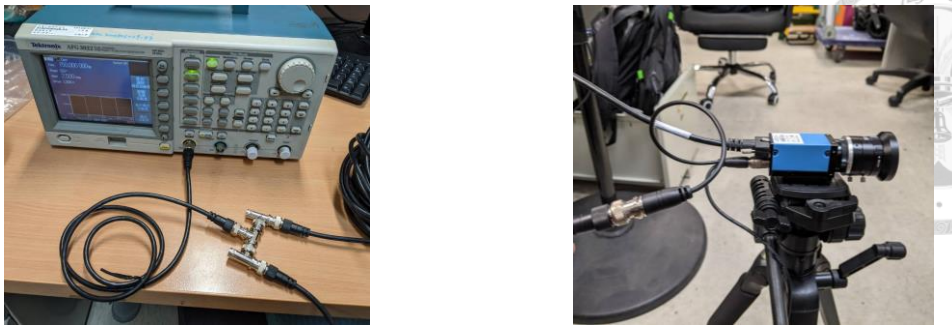


Figure 4.2: System overview of the multi-view system

The overview of the multi-view system is illustrated in Figure 4.2. To control the multiple cameras in the system, the cameras are connected to the computer as the control center. The computer involves the GigE PoE interface card to receive data and supply the power to the cameras through the ethernet cables. Furthermore, to synchronize the capture time for each camera, a signal generator outputs a 0 - 5V squared wave as the trigger signal and connects to all cameras with BNC cables. The connection is shown in Figure 4.3. In the trigger mode, those cameras would capture the images only when the trigger signal is on. Therefore, the maximum frequency of the trigger signal is the same as the maximum sample rate of the cameras. That's equal to 300Hz.

doi:10.6342/NTU202202468



(a) (b)

Figure 4.3: The practical connection of the multi-view system
(a) Connection at the signal generator (b) Connection at the camera

After connecting the whole system, our portable motion capture system is shown in Figure 4.4. There are four cameras capturing the views from different angles. The target human would act in the center area of the system so that the whole human body can be captured in every camera.

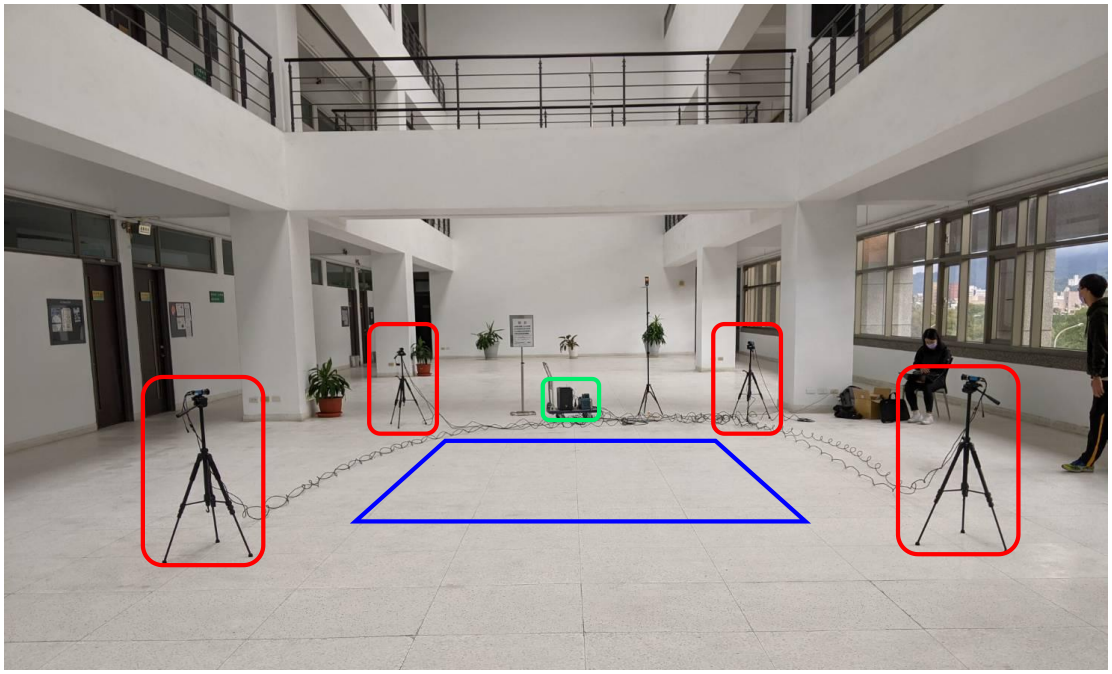


Figure 4.4: The multi-view system in practice
The cameras are framed in red rectangles. The computer and signal generator are framed in green rectangle. The activity range for human motion capture is labeled with blue rectangle.

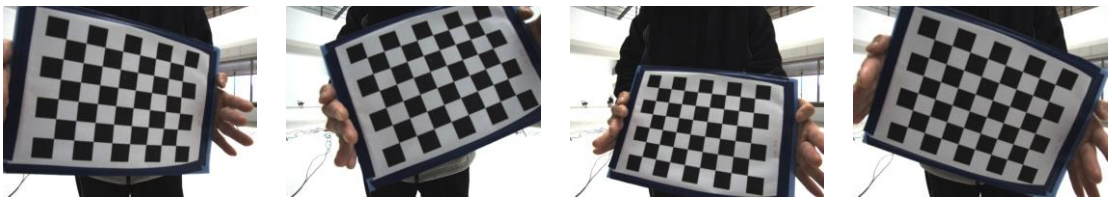
doi:10.6342/NTU202202468

4.2 Camera Calibration and Synchronization

After the multi-view system is built up, camera calibration and synchronization are needed for accurate motion estimation. The camera calibration process is shown in Section 4.2.1. Then, the camera synchronization process is presented in Section 4.2.2.

4.2.1 Camera Calibration

The camera calibration consists of two parts, intrinsic calibration and extrinsic calibration. For intrinsic calibration, Zhang's camera calibration method [54: Zhang 2000] is adopted. As Figure 4.5 shows, the chessboard is captured from several different angles. With the OpenCV [83: Bradski 2022] function, the chessboard corners are detected and would be used to calculate the intrinsic parameters for each camera.



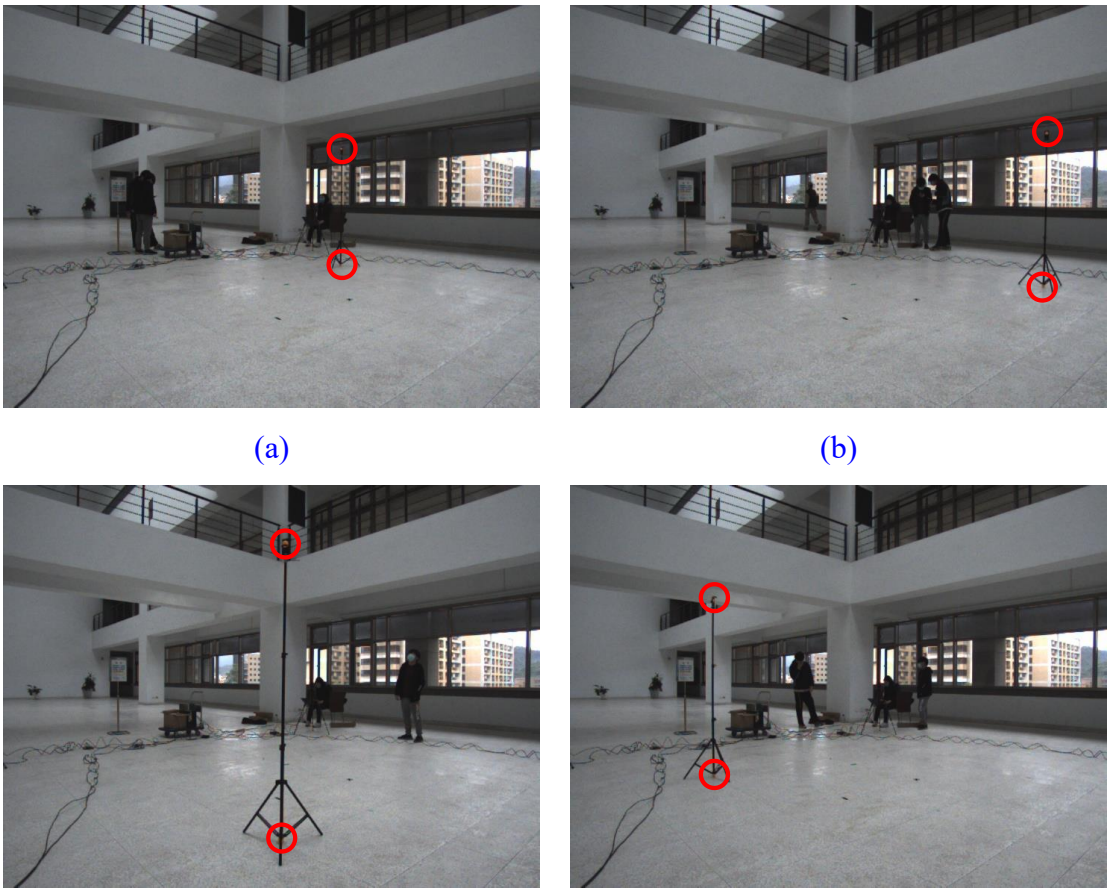
(a) (b) (c) (d)

Figure 4.5: The chessboard images for intrinsic calibration with Zhang's method [54: Zhang 2000]

For extrinsic calibration, since our system is a multi-camera system, every camera should be calibrated to the same world coordinate. Thus, we deposit eight reference points to form a cuboid involving the capture space. With knowing the accurate positions of the reference points, once there are at least six noncoplanar points in the FOV of every camera, the extrinsic parameters can be regarded as a PnP problem and solved with the OpenCV functions.

doi:10.6342/NTU202202468

Another property of this extrinsic calibration method is that the maximum calibration error would occur at the reference points since the cuboid formed by the reference points is a convex set. Therefore, the calibration error in the cuboid can be interpolated with the errors at the reference points. In other words, we can guarantee the upper bound of the calibration error in the activity space while the nonlinear distortion is negligible.



(a) (b)

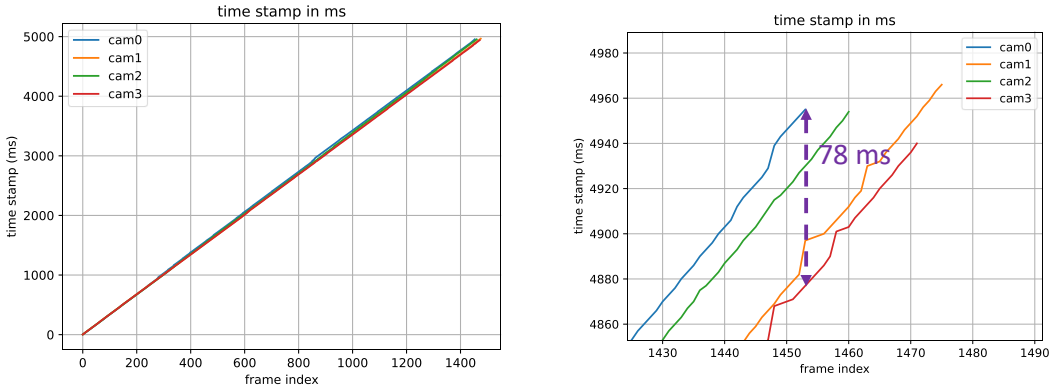
(c) (d)

Figure 4.6: The reference points for camera extrinsic calibration
(a) Rod position 0 (b) Rod position 1 (c) Rod position 2 (d) Rod position 3

doi:10.6342/NTU202202468

4.2.2 Camera Synchronization

To synchronize the captured images from different cameras, the hardware trigger signal is utilized. Despite that, the captured images are still not fully synchronized. In Figure 4.7, there are the time stamps of the frames captured by the four cameras. Ideally, each camera should record 1500 frames in 5 seconds while the trigger single is 300 Hz. However, due to the environment brightness and other hardware issues, there exist 20 – 50 dropped frames for every camera. If we directly use the frame index as the synchronizing label, the synchronization error may be high to 78 milliseconds. This synchronization error may cause huge position errors during the 3D reconstruction for high-speed motions like pitching and batting.



(a) (b)

Figure 4.7: The recorded time stamps for each camera with synchronized hardware trigger (300 Hz in 5 s)
(a) in original scale (b) zoomed in at the end of the recording

doi:10.6342/NTU202202468

As the observation in Figure 4.8, the time difference of the time stamps for each camera didn't drift synchronously. In addition, the time differences are sometimes lower than 3 milliseconds, which is not possible for 300-Hz capturing frequency ideally even if there are dropped frames. Therefore, the problem for hardware triggering is not only the dropped frame issue but also the other hardware problems.

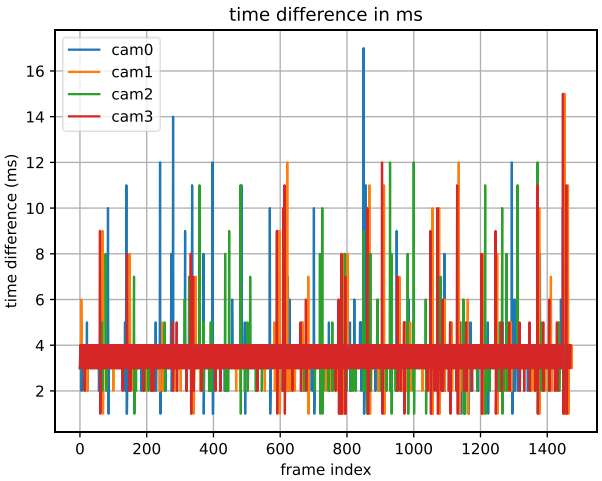


Figure 4.8: The time differences for each camera with synchronized hardware trigger

Not only the synchronization would affect the precision of 3D reconstruction, but the time difference maintains or not would also influence the estimation after post-processes. Therefore, to solve the issues above, we use the ideal time stamps as the benchmark to be matched by the real time stamps from the cameras. The ideal time stamps are generated with the recording frame per second (FPS) as Equation (4.1) where x is the frame index for the output synchronized videos, and y is the ideal time stamps.

$$y = \frac{1000\ ms}{FPS} \cdot x \qquad (4.1)$$

$$\forall x = 1, 2, 3, \ldots$$

doi:10.6342/NTU202202468

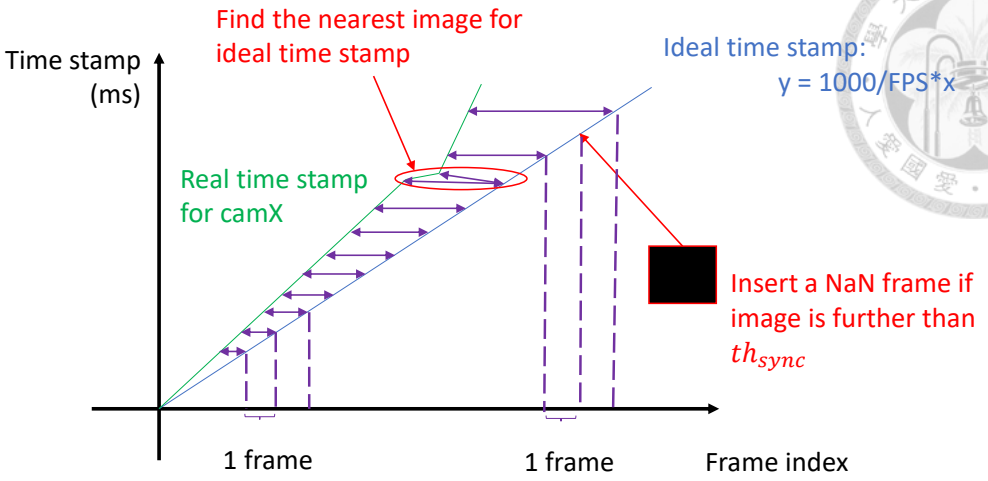


Figure 4.9: The concept of synchronization with time stamps

With the ideal time stamps, we can pick up the nearest frame in one image sequence for every ideal time stamp as the synchronized image sequence, as shown in Figure 4.9. If the nearest frame is further than a threshold th_{sync} (usually equal to 2 or 3 milliseconds) from the ideal time stamp, the synchronized image sequence will be inserted with a NaN frame (fully black image) to prevent the number of frames in the synchronized image sequences not the same for every camera.

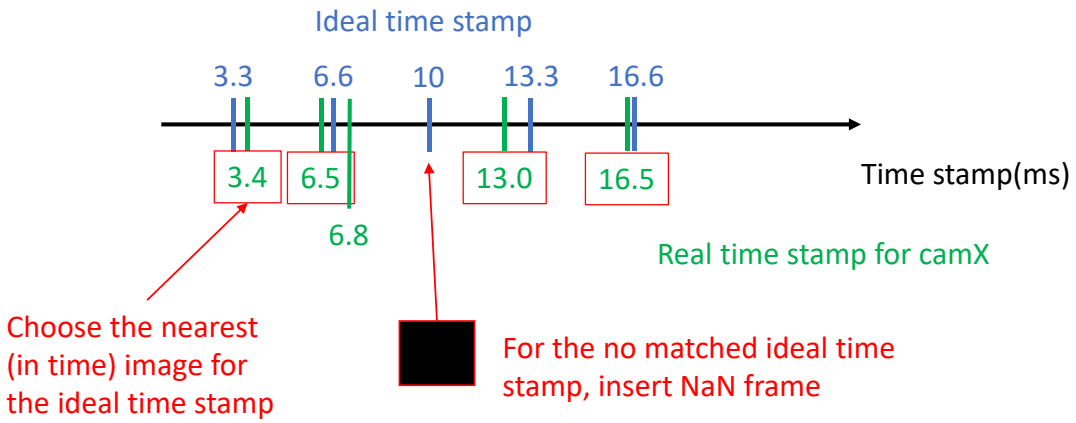


Figure 4.10: An example for synchronization with time stamps ($th_{sync} = 2ms$)

doi:10.6342/NTU202202468

In Figure 4.10, here is an example to present how to output a synchronized image sequence with the ideal time stamps and the real time stamps from a camera. As Figure 4.10 shown, the blue graduations are the ideal time stamps with 300 Hz. Then, the green graduations are the real time stamps captured with a camera “camX”. Similar to the real situation we meet, the sampled times are not exactly the same as the ideal setting, and the sampled frequency is also drifting. To generate the synchronized image sequence, the nearest image would be searched for each ideal time stamp. For the ideal time stamp equal to 10 ms, since the nearest image is 3 ms far from it and th_{sync} is 2 ms, the synchronized image sequence would be inserted a NaN frame at 10 ms. Thus, all the images in the synchronized image sequence are framed with the red rectangles, and the synchronized image sequence can be generated.

doi:10.6342/NTU202202468

4.3 3D Reconstruction from AlphaPose

(a) (b)

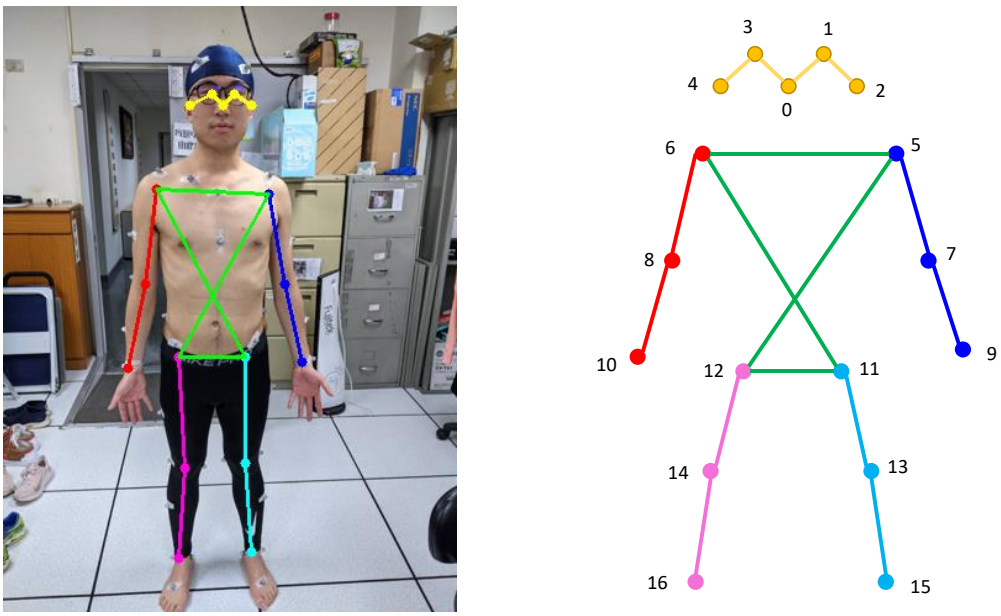
Figure 4.11: 2D target skeleton extraction
(a) Multiple detected 2D skeletons by AlphaPose
(b) Extracted 2D target skeleton

With the camera calibration and the synchronization, the captured multi-view videos are ready to reconstruct the 3D motions. In this stage, AlphaPose [48: Fang et al. 2017] is utilized to extract the 2D keypoints from the images of every view. As mentioned in Section 3.1, AlphaPose is a multi-person 2D human skeleton estimator. During recording, there may be some pedestrians passing through the camera views and detected by AlphaPose as Figure 4.11(a) shows. For our algorithm, it's designed for single-person skeleton estimation. Therefore, the other non-target 2D skeletons would influence our results.

To eliminate those 2D skeletons and extract the target automatically, we first initialize the biggest 2D skeleton in the images as the target skeleton. During the motion, there are two conditions to determine whether the 2D skeleton is the target or not. The first one is that the position differences of the main body (the average position of

doi:10.6342/NTU202202468

shoulders and hips) between any adjacent frames should be under a distance threshold (50 pixels). Secondly, the ratio of the width and height of the bounding boxes should not vary over a set threshold (equal to 1.6) with respect to the target skeleton in the previous frame. The first condition can reject the other skeletons which is far from the target. Then, the second condition can remove the skeletons which have different poses from the target at a near distance. After these two extraction conditions, the 2D target skeleton can be extracted as shown in Figure 4.11(b).



(a) (b)

Figure 4.12: Keypoints on a 2D skeleton
(a) Keypoint positions on human body
(b) Keypoint Indexes

The extracted 2D target skeleton has 17 keypoints as shown in Figure 4.12. There are 5 keypoints on the head, 3 keypoints on each limb. The precise definitions for every keypoint are presented in Table 4.2.

doi:10.6342/NTU202202468

Table 4.2

Keypoint Definition for Microsoft COCO keypoint task from [51: Lin et al. 2015]

Keypoint Index	Point Definition in Human Body
0	Nose
1	Left Eye (LEye)
2	Right Eye (REye)
3	Left Ear (LEar)
4	Right Ear (REar)
5	Left Shoulder (LShoulder)
6	Right Shoulder (RShoulder)
7	Left Elbow (LElbow)
8	Right Elbow (RElbow)
9	Left Wrist (LWrist)
10	Right Wrist (RWrist)
11	Left Hip (LHip)
12	Right Hip (RHip)
13	Left Knee (LKnee)
14	Right Knee (RKnee)
15	Left Ankle (LAnkle)
16	Right Ankle (RAnkle)

After the 2D target skeleton extraction, the next step is to reconstruct the 3D skeleton. As shown in Figure 4.13, the 3D reconstruction process contains three modules: triangulation, reprojection, and calculating reprojection error.

doi:10.6342/NTU202202468

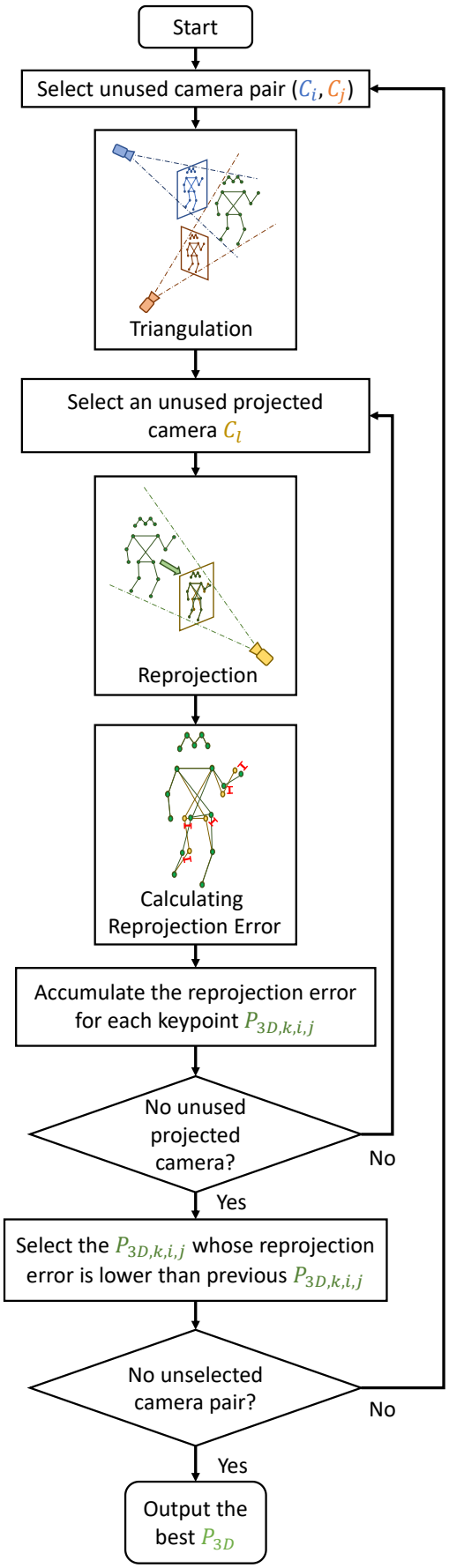
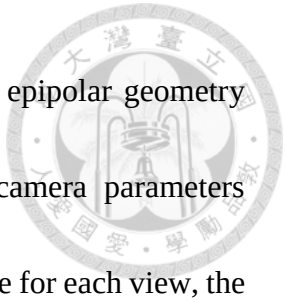


Figure 4.13: Flowchart of 3D Reconstruction from multi-view data

doi:10.6342/NTU202202468



To build the 3D points from 2D points in different views, the epipolar geometry mentioned in Section 3.2 is applied in triangulation. With the camera parameters calibrated in Section 4.2.1 and the 2D skeletons detected by AlphaPose for each view, the epipolar geometry can reconstruct the 3D keypoints in skeletons by selecting any two unrepeated cameras C_i and C_j. However, if the number of cameras is greater than two, there will be more than one reconstructed 3D skeleton. To determine which reconstructed 3D point is optimum for each keypoint, all the 3D skeletons are re-projected to every view C_l and compared with the 2D skeleton generated in that view. For every 3D point in every 3D skeleton, their reprojection errors calculated by the 2-norm distance with the corresponding keypoints in 2D skeletons are accumulated for every camera view. After the traversal, the 3D keypoints whose accumulated reprojection errors are the lowest would be selected to build the optimal 3D skeleton. The overall algorithm is written in **Algorithm 4.1**.

doi:10.6342/NTU202202468

Algorithm 4.1: 3D raw skeleton reconstruction

Input: all 2D keypoint positions of the target human skeleton in C camera views $P_{2D,i}$, camera parameters for each camera M_i

Output: a 3D raw skeleton of the target human P_{3D}

K ← the number of keypoints for a skeleton
$P_{3D} \leftarrow \phi$
for $k \leftarrow 1$ to K **do**
    $e_{sum,min} \leftarrow \infty$
    for $i \leftarrow 1$ to $C-1$ **do**
        for $j \leftarrow i+1$ to C **do**
            $P_{3D,k,i,j} \leftarrow triangulation(P_{2D,k,i}, P_{2D,k,j}, M_i, M_j)$
            $e_{sum} \leftarrow 0$
            for $l \leftarrow 1$ to C **do**
                $\hat{P}_{2D,k,l} \leftarrow project(P_{3D,k,i,j}, M_l)$
                $e_{sum} \leftarrow e_{sum} + \left\| \hat{P}_{2D,k,l} - P_{2D,k,l} \right\|_2$
            end for
            if $e_{sum} < e_{sum,min}$ **do**
                $e_{sum,min} \leftarrow e_{sum}$
                $P_{3D,k} \leftarrow P_{3D,k,i,j}$
            end if
        end for
    end for
    $P_{3D} \leftarrow P_{3D} \cup P_{3D,k}$
end for
return P_{3D}

Although the outputted 3D skeleton is optimal among all triangulated skeletons, there are several shortcomings needed to improve in the reconstructed skeletons. To emphasize the data roughness, we will call the reconstructed 3D skeleton as RawSK (raw skeleton).

The first detection of the RawSKs is that it includes lots of missing data and abnormal keypoint positions (outliers) as Figure 4.14 shown.

doi:10.6342/NTU202202468

(a) (b)

Figure 4.14: Missing data and outlier issues of RawSKs
(a) Missing data in the RawSK for B21_406 at frame 633
(b) Outlier data in the RawSK for B20_354 at frame 276

The missing data and outlier issues cause by the **occlusion**, **frame-dropping**, and some **imperfect training results of AlphaPose**. These problems are common in the RawSKs directly reconstructed from multi-view AlphaPose results.

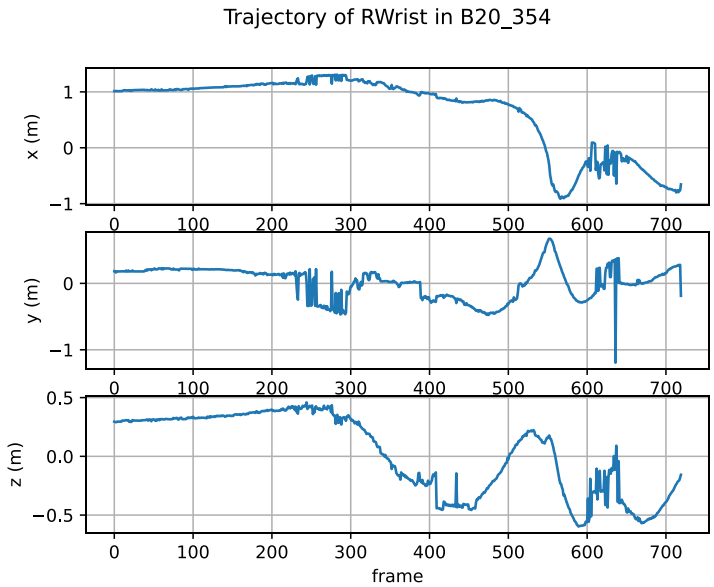


Figure 4.15: The trajectory of right wrist in RawSK of the example

Besides, since the RawSKs are reconstructed frame by frame, the reconstructed positions are independent of the results in other frames. This reason makes the RawSKs

doi:10.6342/NTU202202468

perform shakily, as shown in Figure 4.15, and usually have bad estimations for the derivative metrics such as velocity and acceleration.

In addition, the RawSKs are directly reconstructed from the 2D skeletons. Therefore, some 3D geometry constraints for human skeletons are not concerned. For example, the length between the right elbow to the right wrist would be constant. However, as the 3D position for each keypoint is reconstructed on its own, the bone length constraints are not considered. The distance between the right elbow to the right wrist is varying in the motion sequence as shown in Figure 4.16.

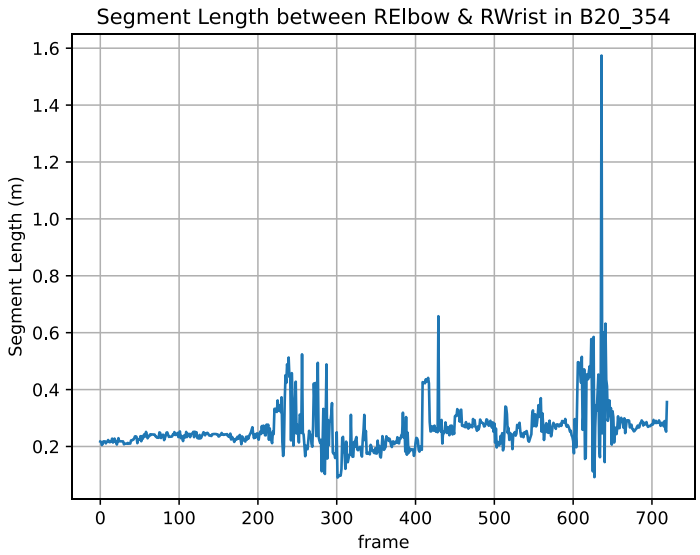


Figure 4.16: The distance between RElbow and RWrist of RawSK in B20_354

With these problems, it's hard to regard the RawSKs as stable and robust motion estimation results. To overcome them, a method modifying the performance of RawSKs with only single-track information is proposed in Chapter 5.

doi:10.6342/NTU202202468

Chapter 5
Single-Track 3D Human Motion Modification

After showing how to reconstruct the 3D raw human pose and the robustness issue for RawSKs in Chapter 4, this chapter would show how to estimate a stable and robust 3D human motion with the proposed method. In Section 5.1, the joint space model for human motion would be described. Secondly, the proposed approach to estimate the body parameters for the joint-space human motion model would be introduced in Section 5.2. Thereafter, a modified UKF to increase the robustness with outlier component rejection and the initialization method would be presented in Section 5.3 and Section 5.4. Finally, an iterative LQR tracking method would be shown to solve the joint coordination problem in Section 5.5.

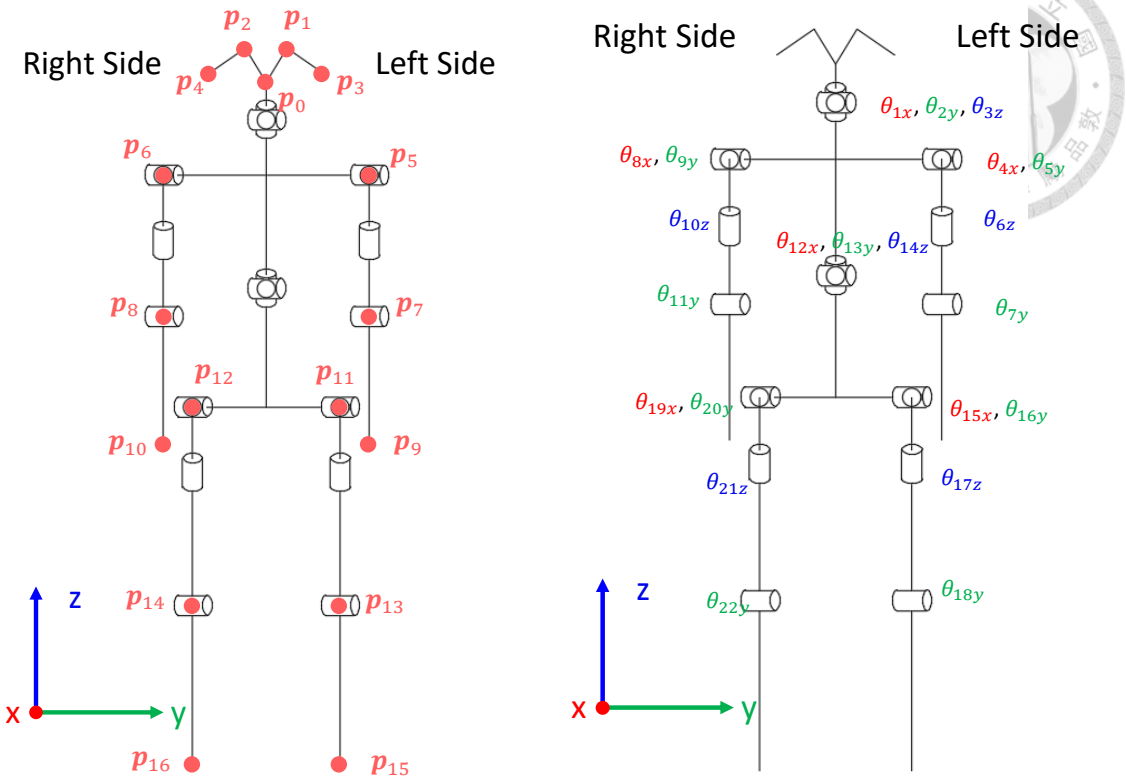
doi:10.6342/NTU202202468

5.1 Kinematic model of Human Skeleton

Referring to the Microsoft COCO keypoint task [51: Lin et al. 2015], the task AlphaPose training for, the keypoint definition is shown in Figure 5.1 (a) and Table 4.2. For a human skeleton, there are 17 keypoints distributd in the whole body, 5 keypoints in the head and 3 keypoints in each limb. If we estimate these 17 keypoints in 3D space, there would be high to 51 degrees of freedom (DOF) needed to be estimated. But, since all the keypoints come from the same human, the motion of every keypoint shouldn't be independent. Considering human motions, we proposed the human motion model in joint space modified from [55: Ude et al. 2004] in Figure 5.1 (b).

There are 22 DOFs to describe the relative motions of the keypoints in a human skeleton. Since the end keypoints of limbs are the wrists and ankles, the 3 DOFs near the end of each limb are ignored because they only affect the points on the hands and feet. Consequentially, the number of DOFs in every limb becomes four from seven for the keypoint definition from Microsoft COCO task. For the spine rotations, it's composed of many small relative motions between the vertebrae. To simplify the kinematic model, we divide the overall spine motion into two groups: neck rotation and spine rotation. In each rotation, there are 3 DOFs including the three-rotating axis in different directions to perform the full rotating DOFs in 3D space.

doi:10.6342/NTU202202468



(a) (b)

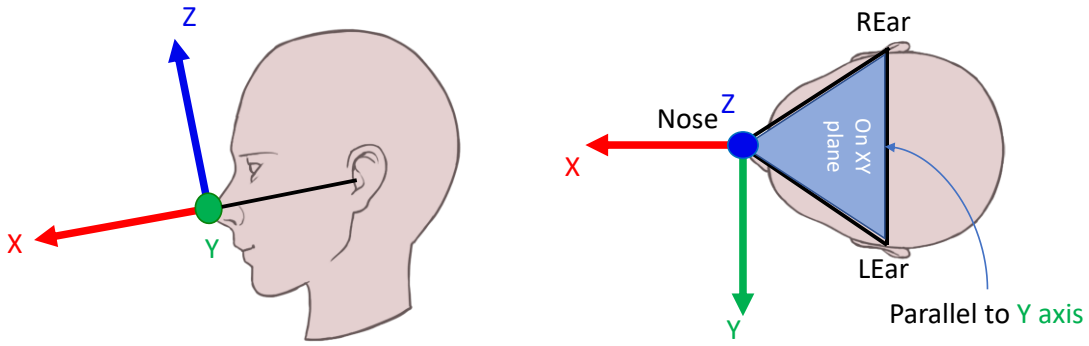
Figure 5.1: Comparison of the representation between Cartesian Space and Joint Space (a) Keypoint Definition in Cartesian Space (b) The Joint Definition for Human Motion

The kinematic model in Figure 5.1 (b) only shows how to describe the relative motions of each keypoints in a human skeleton. However, in Figure 5.1 (a) and our task, the absolute positions of keypoints in the world coordinate is the eventual goal we should estimate. To transform the relative keypoint motions in a human body into the world coordinate, the 6-DOF transformation between the head and world coordinate should also be estimated.

To simply define the head coordinate $\{head\}$, we set the nose position $\boldsymbol{p}_0$ as the origin in $\{head\}$. The vector from REar $\boldsymbol{p}_4$ to LEar $\boldsymbol{p}_3$ is parallel to the y-axis of $\{head\}$. The points of the nose and the ears should on the XY-plane of $\{head\}$. The

doi:10.6342/NTU202202468

illustration is shown in Figure 5.2. Therefore, the states to describe the transformation between the head and world coordinate can be regarded as the position of the nose $\boldsymbol{p}_0$ and the roll α, pitch β, yaw γ angles between the head and the world coordinate.



(a) (b)

Figure 5.2: Head coordinate definition
(a) Side view (b) Top view

Adding the DOFs in the human kinematic model and the head-world transformation, we can reduce the number of the estimated DOFs from 51 to 28, which can dramatically decrease the difficulty of 3D human motion estimation.

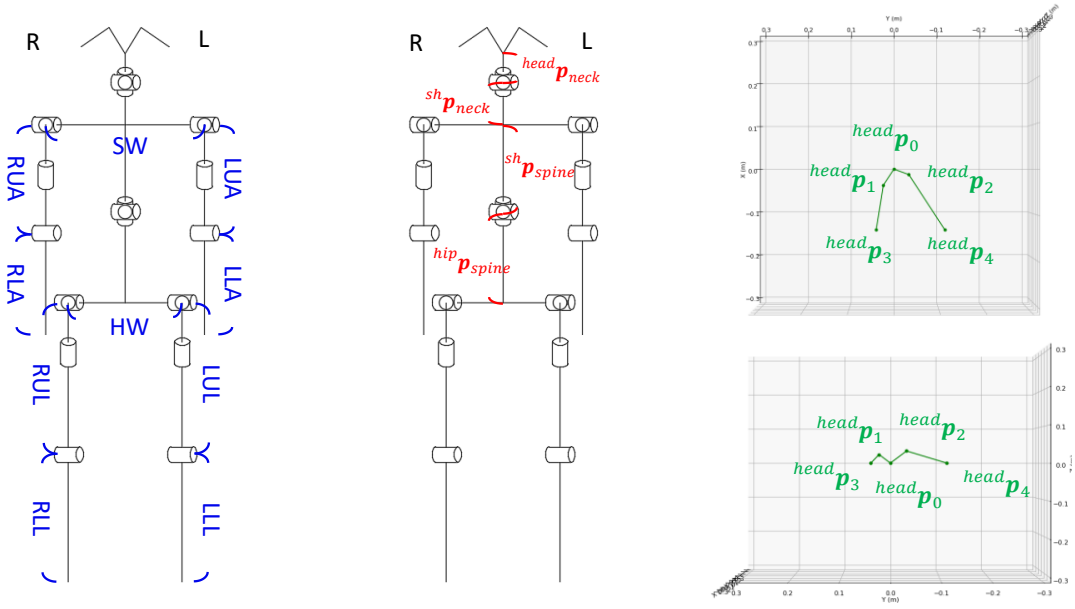
After knowing how many DOFs we should estimate, how exactly the keypoint positions are described with the joint states (22 joint angles + 6 head-world transformation stats) is the next step we concern. To recover the 3D keypoint positions of a human skeleton from the joint space, a forward kinematic function $h(\boldsymbol{x}_{pos})$ is developed in Equation (5.3)-(5.25), where $\boldsymbol{x}_{pos}$ is the state in joint space containing 26 elements shown in Equation (5.1).

doi:10.6342/NTU202202468

$$\boldsymbol{x}_{pos} = [\boldsymbol{\theta}^T \quad \boldsymbol{T}^T]^T \tag{5.1}$$

$$\begin{cases} \boldsymbol{\theta} = [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_{22}]^T \\ \boldsymbol{T} = [p_{0,x} \quad p_{0,y} \quad p_{0,z} \quad \alpha \quad \beta \quad \gamma]^T \end{cases} \tag{5.2}$$

Besides the joint state $\boldsymbol{x}_{pos}$, the body parameters shown in Figure 5.3 and Table 5.1 should also be known before calculating the forward kinematic function $h(\boldsymbol{x}_{pos})$.



(a) (b) (c)

Figure 5.3: Body Parameters defined for Human Skeleton Kinematic Model

(a) Limb parameters

(b) Spine parameters

(c) Head parameters (the keypoint positions in $\{head\}$)

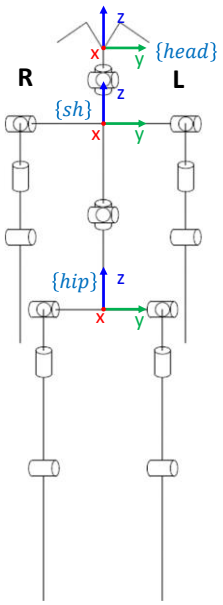
doi:10.6342/NTU202202468

Table 5.1

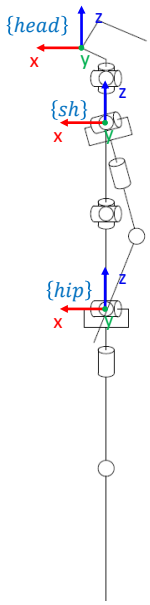
The definition of the body parameters in a human skeleton

Symbol	Definition
SW	Shoulder Width ($\in \mathbb{R}$)
HW	Hip Width ($\in \mathbb{R}$)
LUA	Length of Left Upper Arm ($\in \mathbb{R}$)
LLA	Length of Left Forearm ($\in \mathbb{R}$)
RUA	Length of Right Upper Arm ($\in \mathbb{R}$)
RLA	Length of Right Forearm ($\in \mathbb{R}$)
LUL	Length of Left Thigh ($\in \mathbb{R}$)
LLL	Length of Left Calf ($\in \mathbb{R}$)
RUL	Length of Right Thigh ($\in \mathbb{R}$)
RLL	Length of Right Calf ($\in \mathbb{R}$)
${}^{head}\boldsymbol{p}_{neck}$	Position of Neck rotation center in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{sh}\boldsymbol{p}_{neck}$	Position of Neck rotation center in $\{sh\}$ ($\in \mathbb{R}^3$)
${}^{sh}\boldsymbol{p}_{spine}$	Position of Spine rotation center in $\{sh\}$ ($\in \mathbb{R}^3$)
${}^{hip}\boldsymbol{p}_{spine}$	Position of Spine rotation center in $\{hip\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_0$	Position of Nose in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_1$	Position of LEye in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_2$	Position of REye in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_3$	Position of LEar in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_4$	Position of REar in $\{head\}$ ($\in \mathbb{R}^3$)

doi:10.6342/NTU202202468



(a)



(b)

Figure 5.4: The coordinates defined in the human skeleton
(a) Front view (b) Left side view

For the head keypoints ($\boldsymbol{p}_0$ to $\boldsymbol{p}_4$), to transform them from $\{head\}$ into world coordinate $\{world\}$, the only thing we need is the transformation matrix ${}^{world}_{head}\boldsymbol{T}$, and it can be calculated with Equation (5.1):

$$ {}^{world}_{head}\boldsymbol{R} = \boldsymbol{R}_z(\gamma) \cdot \boldsymbol{R}_y(\beta) \cdot \boldsymbol{R}_x(\alpha) \tag{5.3} $$

$$ {}^{world}_{head}\boldsymbol{t} = [p_{0,x} \quad p_{0,y} \quad p_{0,z}]^T \tag{5.4} $$

$$ {}^{world}_{head}\boldsymbol{T} = \begin{bmatrix} {}^{world}_{head}\boldsymbol{R} & {}^{world}_{head}\boldsymbol{t} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \tag{5.5} $$

The rotation matrices $\boldsymbol{R}_x, \boldsymbol{R}_y, \boldsymbol{R}_z$ follow the definition in Section 3.7.

doi:10.6342/NTU202202468

With the transformation matrix ${}^{world}_{head}T$ and the positions of head keypoints the neck rotation center defined in $\{head\}$, their 3D positions in $\{world\}$ can be calculated as:

$$\begin{bmatrix} {}^{world}\boldsymbol{p}_i \\ 1 \end{bmatrix} = {}^{world}_{head}\boldsymbol{T} \begin{bmatrix} {}^{head}\boldsymbol{p}_i \\ 1 \end{bmatrix}, \qquad \forall i = 0, \ldots, 4 \tag{5.6}$$

$$\begin{bmatrix} {}^{world}\boldsymbol{p}_{neck} \\ 1 \end{bmatrix} = {}^{world}_{head}\boldsymbol{T} \begin{bmatrix} {}^{head}\boldsymbol{p}_{neck} \\ 1 \end{bmatrix} \tag{5.7}$$

For the keypoints in the shoulder coordinate $\{sh\}$ ($\boldsymbol{p}_5$ and $\boldsymbol{p}_6$), we can calculate the transformation matrix ${}^{head}_{sh}\boldsymbol{T}$:

$${}^{head}_{sh}\boldsymbol{R} = \boldsymbol{R}_z(\theta_3)\boldsymbol{R}_y(\theta_2)\boldsymbol{R}_x(\theta_1) \tag{5.8}$$

$${}^{head}_{sh}\boldsymbol{t} = {}^{head}\boldsymbol{p}_{neck} - {}^{head}_{sh}\boldsymbol{R}\,{}^{sh}\boldsymbol{p}_{neck} \tag{5.9}$$

$${}^{head}_{sh}\boldsymbol{T} = \begin{bmatrix} {}^{head}_{sh}\boldsymbol{R} & {}^{head}_{sh}\boldsymbol{t} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \tag{5.10}$$

Then, the positions of $\boldsymbol{p}_5$ and $\boldsymbol{p}_6$ in $\{world\}$ are calculated with the keypoint positions defined in $\{sh\}$:

$$\begin{bmatrix} {}^{world}\boldsymbol{p}_i \\ 1 \end{bmatrix} = {}^{world}_{head}\boldsymbol{T} \cdot {}^{head}_{sh}\boldsymbol{T} \begin{bmatrix} {}^{sh}\boldsymbol{p}_i \\ 1 \end{bmatrix}, \qquad \forall i = 5,6$$
$$\text{where } \begin{cases} {}^{sh}\boldsymbol{p}_5 = [0 \quad SW/2 \quad 0]^T \\ {}^{sh}\boldsymbol{p}_6 = [0 \quad -SW/2 \quad 0]^T \end{cases} \tag{5.11}$$

For the elbow and wrist keypoints, their positions in $\{sh\}$ can be calculated as:

$${}^{sh}p_7 = \boldsymbol{R}_y(\theta_5)\,\boldsymbol{R}_x(\theta_4) \begin{bmatrix} 0 \\ 0 \\ -LUA \end{bmatrix} + {}^{sh}\boldsymbol{p}_5 \tag{5.12}$$

$${}^{sh}\boldsymbol{p}_8 = \boldsymbol{R}_y(\theta_9)\,\boldsymbol{R}_x(\theta_8) \begin{bmatrix} 0 \\ 0 \\ -RUA \end{bmatrix} + {}^{sh}\boldsymbol{p}_6 \tag{5.13}$$

$${}^{sh}\boldsymbol{p}_9 = \boldsymbol{R}_y(\theta_5)\,\boldsymbol{R}_x(\theta_4) \left(\begin{bmatrix} 0 \\ 0 \\ -LUA \end{bmatrix} + \boldsymbol{R}_z(\theta_6)\,\boldsymbol{R}_y(\theta_7) \begin{bmatrix} 0 \\ 0 \\ -LLA \end{bmatrix} \right) + {}^{sh}\boldsymbol{p}_5 \tag{5.14}$$

doi:10.6342/NTU202202468

$$^{sh}\boldsymbol{p}_{10} = \boldsymbol{R}_y(\theta_9)\,\boldsymbol{R}_x(\theta_8)\left(\begin{bmatrix}0\\0\\-RUA\end{bmatrix} + \boldsymbol{R}_z(\theta_{10})\,\boldsymbol{R}_y(\theta_{11})\begin{bmatrix}0\\0\\-RLA\end{bmatrix}\right) + {}^{sh}\boldsymbol{p}_6 \tag{5.15}$$

To transform the elbow and wrist keypoints into $\{world\}$, they can be calculated as:

$$\begin{bmatrix}{}^{world}\boldsymbol{p}_i\\1\end{bmatrix} = {}^{world}_{head}\boldsymbol{T} \cdot {}^{head}_{sh}\boldsymbol{T}\begin{bmatrix}{}^{sh}\boldsymbol{p}_i\\1\end{bmatrix}, \qquad \forall i = 7,\ldots,10 \tag{5.16}$$

For the keypoints in the hip coordinate $\{hip\}$ ($\boldsymbol{p}_{11}$ and $\boldsymbol{p}_{12}$), we can calculate the transformation matrix ${}^{sh}_{hip}\boldsymbol{T}$:

$${}^{sh}_{hip}\boldsymbol{R} = \boldsymbol{R}_z(\theta_{14})\boldsymbol{R}_y(\theta_{13})\boldsymbol{R}_x(\theta_{12}) \tag{5.17}$$

$${}^{sh}_{hip}\boldsymbol{t} = {}^{sh}\boldsymbol{p}_{spine} - {}^{sh}_{hip}\boldsymbol{R}\,{}^{hip}\boldsymbol{p}_{spine} \tag{5.18}$$

$${}^{sh}_{hip}\boldsymbol{T} = \begin{bmatrix}{}^{sh}_{hip}\boldsymbol{R} & {}^{sh}_{hip}\boldsymbol{t}\\ \boldsymbol{0}_{1\times3} & 1\end{bmatrix} \tag{5.19}$$

Then, the positions of $\boldsymbol{p}_{11}$ and $\boldsymbol{p}_{12}$ in $\{world\}$ are calculated with the keypoint positions defined in $\{hip\}$:

$$\begin{bmatrix}{}^{world}\boldsymbol{p}_i\\1\end{bmatrix} = {}^{world}_{head}\boldsymbol{T} \cdot {}^{head}_{sh}\boldsymbol{T} \cdot {}^{sh}_{hip}\boldsymbol{T}\begin{bmatrix}{}^{hip}\boldsymbol{p}_i\\1\end{bmatrix}, \qquad \forall i = 11,12$$
$$where \begin{cases}{}^{hip}\boldsymbol{p}_{11} = [0 \quad HW/2 \quad 0]^T\\ {}^{hip}\boldsymbol{p}_{12} = [0 \quad -HW/2 \quad 0]^T\end{cases} \tag{5.20}$$

For the knee and ankle keypoints, their positions in $\{hip\}$ can be calculated as:

$${}^{hip}\boldsymbol{p}_{13} = \boldsymbol{R}_y(\theta_{16})\,\boldsymbol{R}_x(\theta_{15})\begin{bmatrix}0\\0\\-LUL\end{bmatrix} + {}^{hip}\boldsymbol{p}_{11} \tag{5.21}$$

$${}^{hip}\boldsymbol{p}_{14} = \boldsymbol{R}_y(\theta_{20})\,\boldsymbol{R}_x(\theta_{19})\begin{bmatrix}0\\0\\-RUL\end{bmatrix} + {}^{hip}\boldsymbol{p}_{12} \tag{5.22}$$

$${}^{hip}\boldsymbol{p}_{15} = \boldsymbol{R}_y(\theta_{16})\,\boldsymbol{R}_x(\theta_{15})\left(\begin{bmatrix}0\\0\\-LUL\end{bmatrix} + \boldsymbol{R}_z(\theta_{17})\boldsymbol{R}_y(\theta_{18})\begin{bmatrix}0\\0\\-LLL\end{bmatrix}\right) + {}^{hip}\boldsymbol{p}_{11} \tag{5.23}$$

doi:10.6342/NTU202202468

$$^{hip}\boldsymbol{p}_{16} = \boldsymbol{R}_y(\theta_{20})\,\boldsymbol{R}_x(\theta_{19})\left(\begin{bmatrix}0\\0\\-RUL\end{bmatrix} + \boldsymbol{R}_z(\theta_{21})\boldsymbol{R}_y(\theta_{22})\begin{bmatrix}0\\0\\-RLL\end{bmatrix}\right) + {}^{hip}\boldsymbol{p}_{12} \tag{5.24}$$

To transform the knee and ankle keypoints into $\{world\}$, they can be calculated as:

$$\begin{bmatrix}{}^{world}\boldsymbol{p}_i\\1\end{bmatrix} = {}^{world}_{head}\boldsymbol{T} \cdot {}^{head}_{sh}\boldsymbol{T}\begin{bmatrix}{}^{sh}\boldsymbol{p}_i\\1\end{bmatrix}, \qquad \forall i = 13,\ldots,16 \tag{5.25}$$

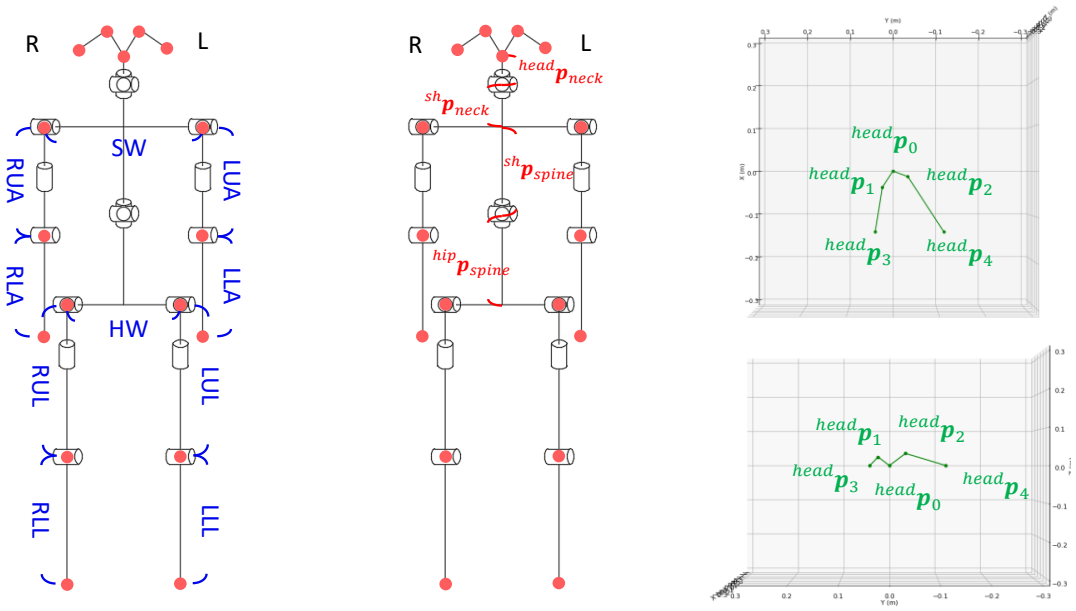
By the combination of Equation (5.3)-(5.25), the forward kinematic function $h(\boldsymbol{x}_{pos})$ can be formed as:

$$\begin{bmatrix}{}^{world}\boldsymbol{p}_0\\{}^{world}\boldsymbol{p}_1\\\vdots\\{}^{world}\boldsymbol{p}_{16}\end{bmatrix} = h(\boldsymbol{x}_{pos}) \in \mathbb{R}^{51} \tag{5.26}$$

doi:10.6342/NTU202202468

5.2 Body parameter estimation

As defined in Table 5.1, there are 19 body parameters needed to be estimated for a human skeleton. As shown in Figure 5.5, the body parameters can be divided into three groups: 1) limb parameters, 2) spine parameters and 3) head parameters. For each group, the estimation methods for the body parameters from the RawSKs are also different.



(a) (b) (c)

Figure 5.5: Body Parameters defined for Human Skeleton Kinematic Model with keypoints positions
(d) Limb parameters
(e) Spine parameters
(f) Head parameters (the keypoint positions in $\{head\}$)

5.2.1 Limb parameter estimation

For limb parameters, they have some common features. First, they are scalar length. Second, both endpoints of limb parameters are the keypoints that can be detected in RawSKs. Because of these two properties, the limb parameters can be estimated easily by calculating the distances between the corresponding keypoints in the RawSKs.

doi:10.6342/NTU202202468

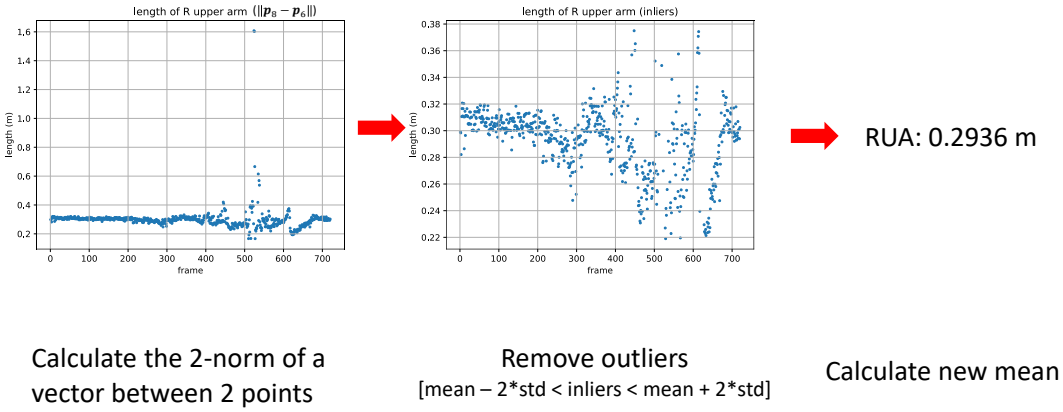


Figure 5.6: The processes for limb parameter estimation

5.2.2 Head parameter estimation

Before the spine parameters, the estimation method for head parameters should be introduced first. The head parameters, in fact, are the head keypoint positions in $\{head\}$. The points on a human's head can be regarded as the points in a rigid body. Their motions should obey the rigid body motion ideally. Therefore, the head parameters are supposed

doi:10.6342/NTU202202468

to be constants. To find these constants, the rigid body transformation ${}^{world}_{head}\boldsymbol{T}$ from head coordinate $\{head\}$ to world coordinate $\{world\}$ should be estimated first.

As the definition of $\{head\}$ in Figure 5.2, we can use Gram-Schmidt process to extract the x-axis unit vector and the y-axis unit vector of $\{head\}$ in $\{world\}$ Then, the z-axis unit vector of $\{head\}$ can be calculated with the cross product for the x-axis unit vector and y-axis unit vector of $\{head\}$. Finally, with the position vector $\boldsymbol{p}_0$ as the translation vector, the ${}^{world}_{head}\boldsymbol{T}$ can be estimated in **Algorithm 5.1**.

Algorithm 5.1: ${}^{world}_{head}\boldsymbol{T}$ estimation

Input: The positions of head keypoints $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_4 \in \mathbb{R}^3$ in $\{world\}$

Output: The transformation matrix ${}^{world}_{head}\boldsymbol{T}$

$\boldsymbol{v}_1 \leftarrow \boldsymbol{p}_3 - \boldsymbol{p}_4$

$\boldsymbol{v}_2 \leftarrow \boldsymbol{p}_0 - (\boldsymbol{p}_3 + \boldsymbol{p}_4)/2$

$\vec{\boldsymbol{y}}_{head} \leftarrow \boldsymbol{v}_1/\|\boldsymbol{v}_1\|$

$\boldsymbol{v}_3 \leftarrow \boldsymbol{v}_2 - (\boldsymbol{v}_2{}^T\vec{\boldsymbol{y}}_{head}) \cdot \vec{\boldsymbol{y}}_{head}$

$\vec{\boldsymbol{x}}_{head} \leftarrow \boldsymbol{v}_3/\|\boldsymbol{v}_3\|$

$\vec{\boldsymbol{z}}_{head} \leftarrow \vec{\boldsymbol{x}}_{head} \times \vec{\boldsymbol{y}}_{head}$

${}^{world}_{head}\boldsymbol{R} \leftarrow [\vec{\boldsymbol{x}}_{head} \quad \vec{\boldsymbol{y}}_{head} \quad \vec{\boldsymbol{z}}_{head}]$

${}^{world}_{head}\boldsymbol{t} \leftarrow \boldsymbol{p}_0$

${}^{world}_{head}\boldsymbol{T} \leftarrow \begin{bmatrix} {}^{world}_{head}\boldsymbol{R} & {}^{world}_{head}\boldsymbol{t} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix}$

return ${}^{world}_{head}\boldsymbol{T}$

The transformation matrix ${}^{world}_{head}\boldsymbol{T}$ can be estimated for every frame in RawSKs once the keypoints $\boldsymbol{p}_0, \boldsymbol{p}_3, \boldsymbol{p}_4$ are not missed. With ${}^{world}_{head}\boldsymbol{T}$ for every frame, the head keypoints $\boldsymbol{p}_0$ to $\boldsymbol{p}_4$ in $\{head\}$ can be calculated by pre-multiplying ${}^{world}_{head}\boldsymbol{T}^{-1}$.

To estimate the head parameters, we have the head keypoint positions in $\{head\}$ for every frame in the RawSK. With the implementation of rigid body assumption, the

doi:10.6342/NTU202202468

vectors between any two keypoints should remain the same in every frame. By minimizing the sum of the squared 2-norm of the vector difference between the optimal keypoint vector $\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j$ and the keypoint vector ${}^{head}\boldsymbol{p}_{i,t} - {}^{head}\boldsymbol{p}_{j,t}$ in every frame t, the optimization problem for head parameter estimation is formed in Equation (5.27):

$$\min_{\hat{\boldsymbol{p}}_1,\hat{\boldsymbol{p}}_2,\hat{\boldsymbol{p}}_3,\hat{\boldsymbol{p}}_4} \sum_{t=0}^{N-1} \sum_{i=0}^{3} \sum_{j=i+1}^{4} \left(\left\| \left({}^{head}\boldsymbol{p}_{i,t} - {}^{head}\boldsymbol{p}_{j,t} \right) - \left(\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j \right) \right\|_2 \right)^2 \tag{5.27}$$

With the proof shown in Appendix A, the optimal solution for Equation (5.27) is:

$$\hat{\boldsymbol{p}}_i = \hat{\boldsymbol{p}}_{\boldsymbol{0}} + \frac{1}{N} \sum_{t=0}^{N-1} {}^{head}\boldsymbol{p}_{i,t} - {}^{head}\boldsymbol{p}_{0,t}\,, \qquad \forall i = 1,\ldots,4 \tag{5.28}$$

Since $\hat{\boldsymbol{p}}_{\boldsymbol{0}}$ is set as $[0 \quad 0 \quad 0]^T$ in $\{head\}$, the head parameters can be estimated as:

$${}^{head}\boldsymbol{p}_i = \hat{\boldsymbol{p}}_i = \frac{1}{N} \sum_{t=0}^{N-1} {}^{head}\boldsymbol{p}_{i,t} - {}^{head}\boldsymbol{p}_{\boldsymbol{0},t}\,, \qquad \forall i = 1,\ldots,4 \tag{5.29}$$

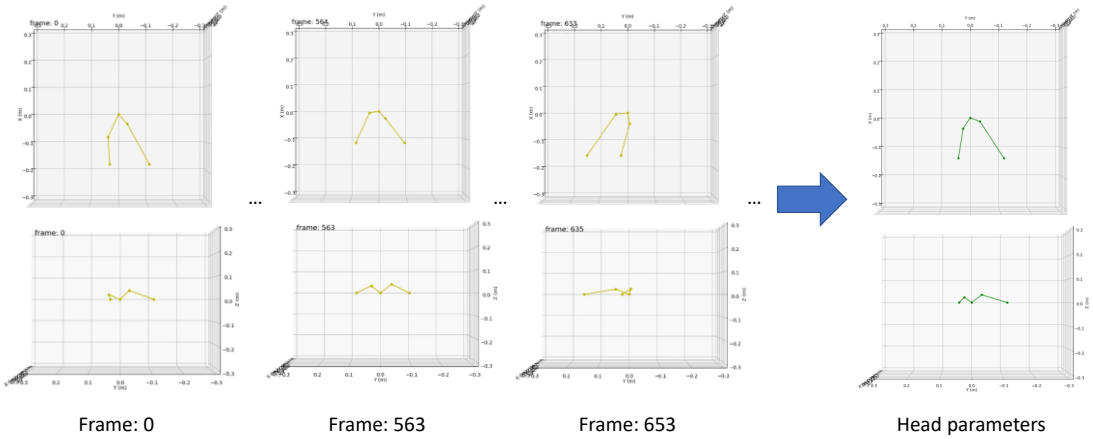


Figure 5.7: The head keypoints in $\{head\}$ for RawSK and the estimated head parameters

doi:10.6342/NTU202202468

5.2.3 Spine parameter estimation

For each spine parameter, one of its endpoints is not included in the RawSKs. Therefore, the direct estimation method for the limb parameter doesn't suit the spine parameter. By the observation of the kinematic model, we can find that both the neck rotation joint and spine rotation joint are spherical joints. Thus, while observing in a coordinate fixed in one link of the spherical joint, the motion of the other endpoint should be on a spherical surface. The center of the spherical surface is the position of the spherical joint, and its radius shows the length of the second link, illustrated in Figure 5.8.

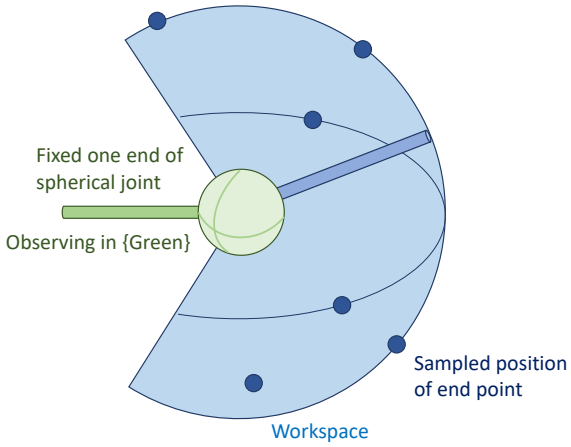
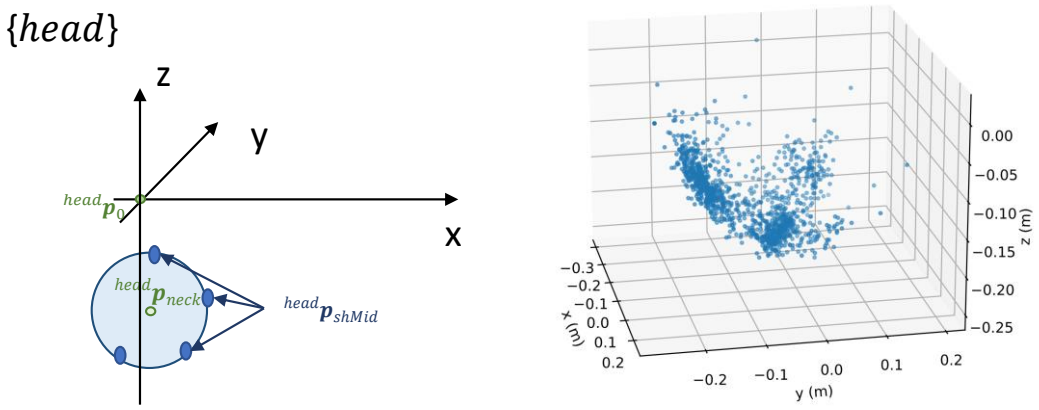


Figure 5.8: The schematic diagram of the motion of the end point while the other end point is fixed for spherical joint

For the neck spherical joint, we can fix our observation in $\{head\}$ by transforming the keypoints in $\{world\}$ with ${}^{world}_{head}\boldsymbol{T}^{-1}$ estimated in Section 5.2.2. To estimate ${}^{head}\boldsymbol{p}_{neck}$ and ${}^{sh}\boldsymbol{p}_{neck}$, the middle point of shoulders ($\boldsymbol{p}_5$ and $\boldsymbol{p}_6$) is chosen as the

doi:10.6342/NTU202202468

observed point, denoted as ${}^{head}\boldsymbol{p}_{shMid}$ in $\{head\}$. The positions of ${}^{head}\boldsymbol{p}_{shMid}$ at different time are shown in Figure 5.9.



(a) (b)

Figure 5.9: The schematic and scatter diagram of ${}^{head}\boldsymbol{p}_{shMid}$
(a) the sphere fitting to estimate ${}^{head}\boldsymbol{p}_{neck}$ and ${}^{sh}\boldsymbol{p}_{neck}$
(b) the scatter diagram of ${}^{head}\boldsymbol{p}_{shMid}$

With the positions of the shoulder middle point at different time, we can use sphere fitting in Section 3.3 to find the rotation center of neck in $\{head\}$, i.e. ${}^{head}\boldsymbol{p}_{neck}$.

$$\boldsymbol{c}_{neck}, r_{neck} \leftarrow SphereFitting\left({}^{head}\boldsymbol{p}_{shMid,t}\right) \tag{5.30}$$

$${}^{head}\boldsymbol{p}_{neck} = \boldsymbol{c}_{neck} \tag{5.31}$$

Besides, the estimated radius of the sphere also represents the distance between the shoulder middle point and the neck rotation center. By defining the z-axis of $\{sh\}$ parallel to the vector from the shoulder middle point to the neck rotation center, ${}^{sh}\boldsymbol{p}_{neck}$ can be estimated as:

$${}^{sh}\boldsymbol{p}_{neck} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ r_{neck} \end{bmatrix} \tag{5.32}$$

doi:10.6342/NTU202202468

Similar to the neck rotation center, the spine parameters ${}^{sh}\boldsymbol{p}_{spine}$ and ${}^{hip}\boldsymbol{p}_{spine}$ can also be estimated with the sphere fitting of the middle point of hips ($\boldsymbol{p}_{11}$ and $\boldsymbol{p}_{12}$) in $\{sh\}$.

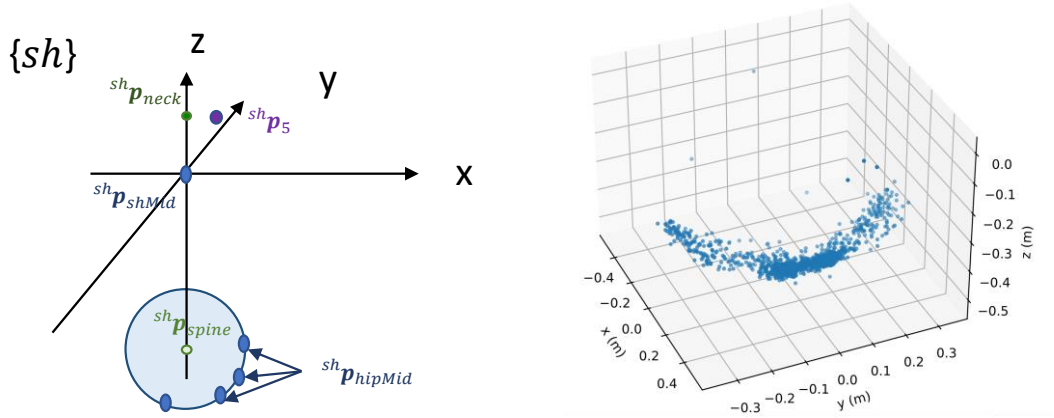
$$\boldsymbol{c}_{spine}, r_{spine} \leftarrow SphereFitting\left({}^{sh}\boldsymbol{p}_{hipMid,t}\right) \tag{5.33}$$

$${}^{sh}\boldsymbol{p}_{spine} = \boldsymbol{c}_{spine} \tag{5.34}$$

In addition, we set the z-axis of $\{sh\}$ parallel to the vector from shoulder middle point the neck rotation center. Therefore, ${}^{hip}\boldsymbol{p}_{spine}$ can be estimated as:

$${}^{hip}\boldsymbol{p}_{spine} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ r_{spine} \end{bmatrix} \tag{5.35}$$

The example of spine parameters near spine rotation center is shown in Figure 5.10.



(a) (b)

Figure 5.10: The schematic and scatter diagram of ${}^{sh}\boldsymbol{p}_{hipMid}$
(a) the sphere fitting to estimate ${}^{sh}\boldsymbol{p}_{spine}$ and ${}^{hip}\boldsymbol{p}_{spine}$
(b) the scatter diagram of ${}^{sh}\boldsymbol{p}_{hipMid}$

doi:10.6342/NTU202202468

5.3 Modified UKF with Outlier Component Rejection and State Constraints

5.3.1 UKF implementation for human motion estimation

With the skeleton kinematic model and the body parameter estimation, the 3D human motion estimation problem becomes a joint stat estimation problem. In control theory, the observers are common and useful tools for state estimation. Kalman filter is one of the most famous observers by minimizing the estimated state covariance. However, since the human skeleton is a highly nonlinear system, a variation of Kalman filter, unscented Kalman filter (UKF), is more suitable for our problem.

Referring to Section 3.4, the UKF can be formulated as:

$$\begin{cases} \boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) + \boldsymbol{w}_t \\ \boldsymbol{z}_t = h(\boldsymbol{x}_t) + \boldsymbol{v}_t \end{cases} \tag{5.36}$$

$$\begin{cases} \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_{UKF}) \\ \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_{UKF}) \end{cases} \tag{5.37}$$

The state transition function $f(\cdot)$ is set as the constant velocity model, i.e.:

$$f(\boldsymbol{x}_{t-1}, \boldsymbol{u}_t) = \boldsymbol{A}_{UKF}\boldsymbol{x}_{t-1} + \boldsymbol{B}_{UKF}\boldsymbol{u}_t \tag{5.38}$$

The $\boldsymbol{A}_{UKF}$ and $\boldsymbol{B}_{UKF}$ are:

$$\boldsymbol{A}_{UKF} = \begin{bmatrix} \boldsymbol{I}_{22} & \boldsymbol{I}_{22} & \boldsymbol{0}_{22\times6} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{22\times22} & \boldsymbol{I}_{22} & \boldsymbol{0}_{22\times6} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times22} & \boldsymbol{I}_6 & \boldsymbol{I}_6 \\ \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times6} & \boldsymbol{I}_6 \end{bmatrix} \in \mathbb{R}^{56\times56} \tag{5.39}$$

$$\boldsymbol{B}_{UKF} = \begin{bmatrix} \boldsymbol{0}_{22\times22} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{I}_{22} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times6} \\ \boldsymbol{0}_{6\times22} & \boldsymbol{I}_6 \end{bmatrix} \in \mathbb{R}^{56\times28} \tag{5.40}$$

doi:10.6342/NTU202202468

The estimated state $\boldsymbol{x}_t$ and the input $\boldsymbol{u}_t$ are:

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{\theta}_t^T & \boldsymbol{\theta}_{\Delta,t}^T & \boldsymbol{T}_t^T & \boldsymbol{T}_{\Delta,t}^T \end{bmatrix}^T \in \mathbb{R}^n, \qquad n = 56 \tag{5.41}$$

$$\boldsymbol{u}_t = \begin{bmatrix} \boldsymbol{\theta}_{\Delta^2,t}^T & \boldsymbol{T}_{\Delta^2,t}^T \end{bmatrix}^T \in \mathbb{R}^{n/2} \tag{5.42}$$

Not only the position state $\boldsymbol{\theta}_t$ and $\boldsymbol{T}_t$, but the discrete velocity terms $\boldsymbol{\theta}_{\Delta,t}$ and $\boldsymbol{T}_{\Delta,t}$ are also included in the state $\boldsymbol{x}_t$. Correspondingly, the input $\boldsymbol{u}_t$ for the constant velocity model is the discrete acceleration $\boldsymbol{\theta}_{\Delta^2,t}$ and $\boldsymbol{T}_{\Delta^2,t}$. However, as there is no acceleration measurement in our multi-view motion capture system, the state transition function $f(\cdot)$ for our estimation becomes:

$$f(\boldsymbol{x}_{t-1}) = \boldsymbol{A}_{UKF}\boldsymbol{x}_{t-1} \tag{5.43}$$

For the observation function $h(\boldsymbol{x}_t)$, it has the same output with the forward kinematic function in Equation (5.26), while the input is $\boldsymbol{x}_t$ rather than $\boldsymbol{x}_{pos}$.

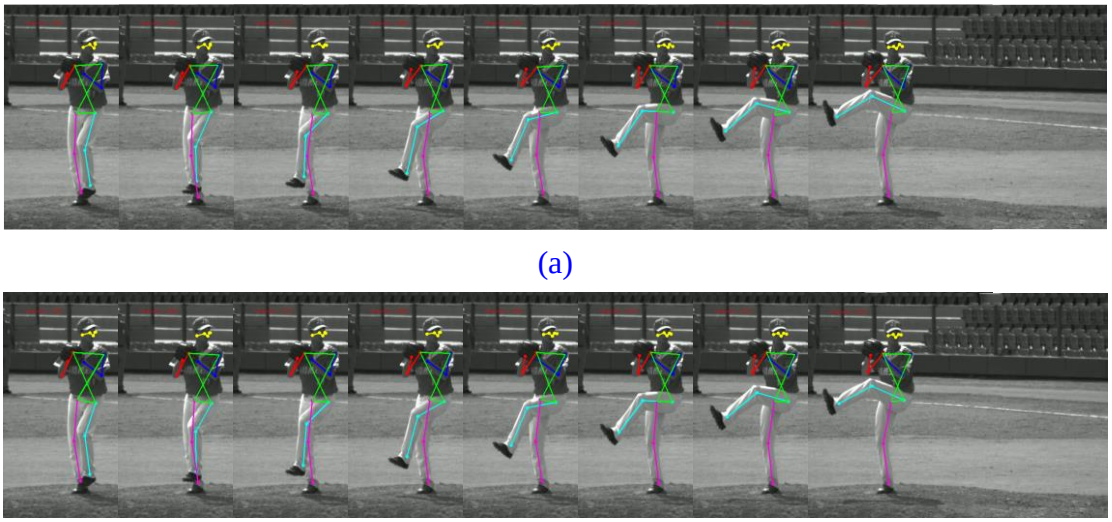
$$\boldsymbol{z}_t = \begin{bmatrix} {}^{world}\boldsymbol{p}_0 \\ {}^{world}\boldsymbol{p}_1 \\ \vdots \\ {}^{world}\boldsymbol{p}_{16} \end{bmatrix} = h(\boldsymbol{x}_t) \in \mathbb{R}^m, \qquad m = 51 \tag{5.44}$$

And the observation $\boldsymbol{z}_t \in \mathbb{R}^{51}$ is the 3D positions of every keypoint.

With the equations above, we can implement the UKF for our joint state estimation with some parameters set by hand. These parameters are: initial state covariance $\boldsymbol{P}_0$, processing noise covariance $\boldsymbol{Q}_{UKF}$, and measurement noise covariance $\boldsymbol{R}_{UKF}$. Besides, the initial state $\hat{\boldsymbol{x}}_0$ is also a parameter we should estimate before running the UKF, which would be introduced in Section 5.4.

doi:10.6342/NTU202202468

With the proper setting of these parameters and set the component of the innovation vector as zero vector for the missed data, the estimation performs well when the observation from RawSKs is stable as Figure 5.11 shows.



(a)

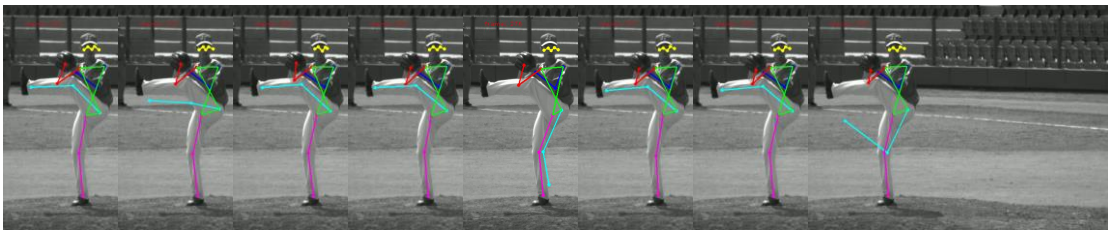
(b)

Figure 5.11: The estimation result with UKF in frame 150-220 for B20_354
(a) the RawSK in frame 150-220 for B20_354
(b) the UKF filtered in frame 150-220 for B20_354

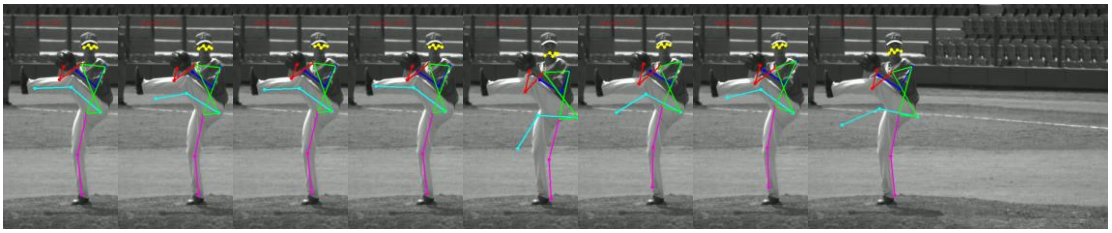
doi:10.6342/NTU202202468

5.3.2 Outlier Component Rejecting UKF (OCR-UKF)

However, when there are few observation outliers with dramatic errors in the RawSKs, the performance would be influenced by the outlier significantly. As shown in Figure 5.12, the outliers occur on the left leg during the lifting process. Despite the fact that the outliers only happen in few separate frames, the filtered result will be affected over a long interval.



(a)



(b)

Figure 5.12: The effect of outliers in RawSK to the UKF filtered result
(a) the RawSK in frame 272-279 for B20_354
(b) the UKF filtered in frame 272-279 for B20_354

To avoid this issue, an **outlier rejection mechanism** is necessary. So far, there are several outlier rejection methods designed for Kalman filter, such as [56: Agamennoni et al. 2011], [57: Ting et al. 2007] and [58: Mu & Yuen 2015]. However, for these methods, the minimum detection size of the outlier detection is a frame, which means that they would regard the whole skeleton in the frame as an outlier when there is only one large error keypoint. For our problem in practice, the number of the outlier keypoints is few,

doi:10.6342/NTU202202468

usually about 0 to 2. If the outlier rejection for a whole frame is adopted, lots of information that comes from the other inlier keypoints will be wasted.

To utilize all the inlier keypoints in every frame, we proposed the **outlier-component-rejecting UKF** (OCR-UKF) to filter out the outliers with the keypoint as the unit. The idea comes from the simple linear regression mentioned in Section 3.5.

For every frame before the measurement update, the UKF would predict the one-step advanced observation covariance $\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}$ as Equation (3.26). In $\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}$, it contains the all predicted observation covariance for each component in innovation vector $\boldsymbol{\varepsilon}_t = \boldsymbol{z}_t - \hat{\boldsymbol{z}}_{t|t-1} \in \mathbb{R}^m$ as:

$$\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1} = \begin{bmatrix} p_{1,1} & \cdots & p_{1,l} & \cdots & p_{1,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{l,1} & \cdots & p_{l,l} & \cdots & p_{l,m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{m,1} & \cdots & p_{m,l} & \cdots & p_{m,m} \end{bmatrix}, \boldsymbol{\varepsilon}_t = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_l \\ \vdots \\ \varepsilon_m \end{bmatrix} \tag{5.45}$$

With the simple linear regression, every innovation component ε_l can be estimated with any innovation component ε_i as:

$$\hat{\varepsilon}_l = \hat{\alpha}_{l.i} + \hat{\beta}_{l.i}\varepsilon_i \tag{5.46}$$

We assume that the probability of innovation vectors is unbiased. Thus, $\hat{\alpha}_{l.i} = \mu_{\varepsilon_l} - \hat{\beta}_{l.i}\mu_{\varepsilon_i} = 0$ with $\mu_{\varepsilon_i} = \mu_{\varepsilon_l} = 0$.

doi:10.6342/NTU202202468

By taking the weighted average of the estimation from all components ε_i, we can estimate the component of the innovation vector $\hat{\varepsilon}_l$ as:

$$\hat{\varepsilon}_l = \frac{1}{\sum_{i=1}^{m} {\rho_{l,i}}^2} \sum_{i=1}^{m} {\rho_{l,i}}^2 \hat{\beta}_{l.i} \varepsilon_i$$

$$= \frac{1}{\sum_{i=1}^{m} {\rho_{l,i}}^2} \sum_{i=1}^{m} {\rho_{l,i}}^2 \frac{p_{l,i}}{p_{i,i}} \varepsilon_i \qquad (5.47)$$

$$\text{while } {\rho_{l,i}}^2 = \frac{{p_{l,i}}^2}{p_{i,i} p_{l,l}} \text{ is the weight}$$

Following the assumption from Kalman filter that the predicted observation distributions are gaussian with mean $\hat{\boldsymbol{z}}_{t|t-1}$ and covariance $\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}$, every innovation vector $\boldsymbol{\varepsilon}_t = \boldsymbol{z}_t - \hat{\boldsymbol{z}}_{t|t-1}$ should also be on the gaussian distribution with zero mean and covariance $\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}$ if the observation $\boldsymbol{z}_t$ follow the same distribution of the predicted observation.

With this assumption, the covariance between every component ε_l and ε_i in $\boldsymbol{\varepsilon}_t$ is $p_{l,i}$. In Section 3.5 and [73: Chatterjee & Hadi 2006], the slope minimizing the sum of squared errors of linear regression result is $\hat{\beta}_{l.i} = \frac{p_{l,i}}{p_{i,i}}$. Therefore, we can use $\hat{\beta}_{l.i}$ to estimate ε_l with any other component ε_i. However, the relationship between ε_l and ε_i may be low or even not relative. It will make the estimated ε_l not meaningful. Thus, the estimated result $\hat{\varepsilon}_l$ comes from the weighing average of each estimation with the weights are the squared correlation coefficients between ε_l and ε_i to emphasize the estimated results from the highly relative components.

doi:10.6342/NTU202202468

Since the estimated component $\hat{\varepsilon}_l$ comes from the estimated observation covariance and the new observation, it would be strange if the estimated component $\hat{\varepsilon}_l$ is far from the real component ε_l. Therefore, a confidence score s_l can be calculated in Equation (5.48) where k_{out} is the outlier coefficient set by hand.

$$s_l = e^{-k_{out}\frac{|\hat{\varepsilon}_l - \varepsilon_l|^2}{p_{l,l}}} \tag{5.48}$$

The range of s_l is $(0,1]$ set by the definition. It can be regarded as an index that measures how possible the innovation component ε_l is an inlier. If $s_l < 0.5$, the innovation component ε_l will be treated as an outlier, vice versa. As $\boldsymbol{\varepsilon}_t = \boldsymbol{z}_t - \hat{\boldsymbol{z}}_{t|t-1}$, the observation component $z_{l,t}$ will be regarded as an outlier component when ε_l is an outlier. Additionally, $z_{l,t}$ will be regarded as an outlier component when the component is missed.

The strictness of the outlier detection can be adjusted by the outlier coefficient k_{out}. When k_{out} increases, the observation component would have smaller tolerance to become an inlier. While k_{out} decreases, the observation component would have fewer chances to become an outlier.

For our problem, as mentioned in Section 4.3, every keypoints in the RawSKs is reconstructed independently. But the positions in each axis are interrelated for a keypoint. Therefore, a keypoint should be treated as the minimum unit for the outlier detection. If

doi:10.6342/NTU202202468

any confidence score of the three axes of the keypoint is smaller than 0.5, the whole keypoint should be regarded as an outlier, as illustrated in Figure 5.13.

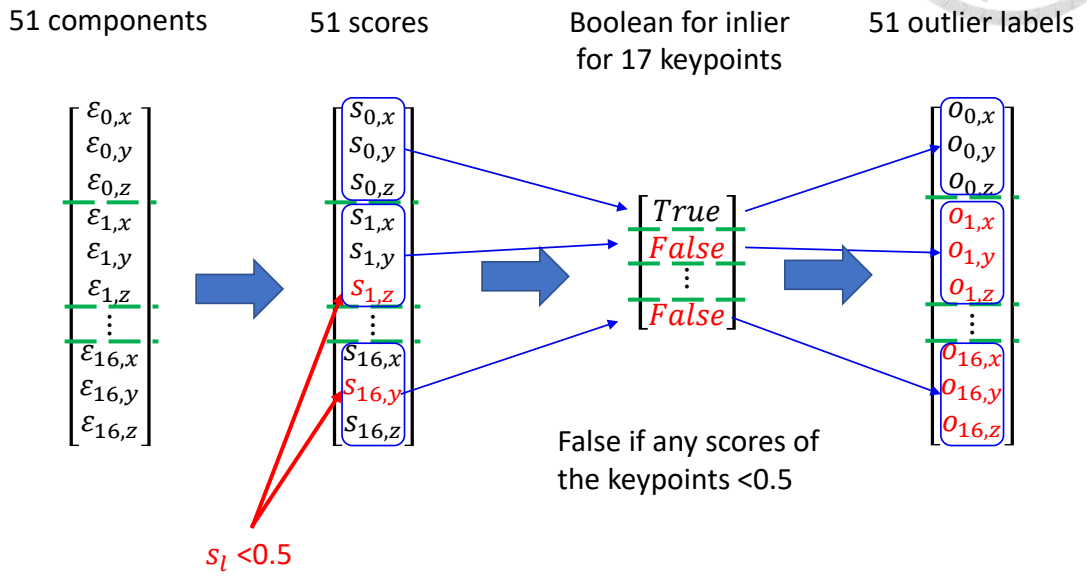


Figure 5.13: The flow to set a keypoint as the minimum unit for outlier detection

Then, an example for outlier detection is shown in Figure 5.14.

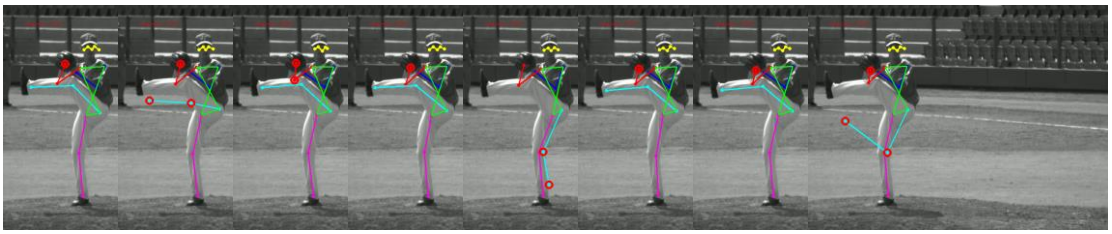


Figure 5.14: Outlier component detection with OCR-UKF in frame 272-279 for B20_354
The outlier keypoints determined by OCR-UKF are marked with red circles.

After the outlier detection, to eliminate the influence of the outliers, the modified Kalman gain $\boldsymbol{K}_{OCR,t}$ for OCR-UKF is formed:

$$\boldsymbol{K}_t = [\boldsymbol{k}_1 \quad \cdots \quad \boldsymbol{k}_l \quad \cdots \quad \boldsymbol{k}_m] \in \mathbb{R}^{n\times m} \tag{5.49}$$

where $\boldsymbol{k}_l \in \mathbb{R}^n$

$$\boldsymbol{o}_l = \begin{cases} \boldsymbol{0}_{n\times n}, & if\ the\ keypoint\ is\ outlier \\ \boldsymbol{I}_n, & if\ the\ keypoint\ is\ inlier \end{cases} \tag{5.50}$$

doi:10.6342/NTU202202468

$$\boldsymbol{K}_{OCR,t} = [\boldsymbol{o}_1\boldsymbol{k}_1 \quad \cdots \quad \boldsymbol{o}_l\boldsymbol{k}_l \quad \cdots \quad \boldsymbol{o}_m\boldsymbol{k}_m] \in \mathbb{R}^{n\times m} \tag{5.51}$$

With the modified Kalman gain, the measurement update for OCR-UKF is turned into:

$$\begin{aligned} \hat{\boldsymbol{x}}_t &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_{OCR,t}\left(\boldsymbol{z}_t - \hat{\boldsymbol{z}}_{t|t-1}\right) \\ &= \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_{OCR,t}\boldsymbol{\varepsilon}_t \\ &= \hat{\boldsymbol{x}}_{t|t-1} + \sum_{l=0}^{m} \boldsymbol{o}_l\boldsymbol{k}_l\varepsilon_l \end{aligned} \tag{5.52}$$

With $\boldsymbol{o}_l$, the outlier component ε_l wouldn't affect the estimated state $\hat{\boldsymbol{x}}_t$ anymore.

However, since the Kalman gain is modified, the covariance update in Equation (3.58) for UKF would not suit for OCR-UKF anymore.

doi:10.6342/NTU202202468

With the update equation for estimated state, for an arbitrary Kalman gain $\boldsymbol{K}_t$, the updated state covariance is:

$$\begin{aligned}
\boldsymbol{P}_t &= cov\left(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t}\right) \\
&= cov(\boldsymbol{x}_t - (\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t)) \\
&= cov\left(\boldsymbol{x}_t - \left[\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\left(\boldsymbol{z}_t - h\left(\hat{\boldsymbol{x}}_{t|t-1}\right)\right)\right]\right) \\
&= cov\left(\boldsymbol{x}_t - \left[\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\left(h(\boldsymbol{x}_t) + \boldsymbol{v}_t - h\left(\hat{\boldsymbol{x}}_{t|t-1}\right)\right)\right]\right) \\
&= cov\left(\left(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}\right) - \boldsymbol{K}_t\left[h(\boldsymbol{x}_t) - h\left(\hat{\boldsymbol{x}}_{t|t-1}\right)\right] - \boldsymbol{K}_t\boldsymbol{v}_t\right) \\
&= cov\left(\left(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}\right) - \boldsymbol{K}_t\left[h(\boldsymbol{x}_t) - h\left(\hat{\boldsymbol{x}}_{t|t-1}\right)\right]\right) + \boldsymbol{K}_t\boldsymbol{R}_{UKF}{\boldsymbol{K}_t}^T
\end{aligned} \tag{5.53}$$

Define a matrix $\boldsymbol{H}_t \in \mathbb{R}^{m\times n}$ such that $\boldsymbol{K}_t\left[h(\boldsymbol{x}_t) - h\left(\hat{\boldsymbol{x}}_{t|t-1}\right)\right] = \boldsymbol{K}_t\boldsymbol{H}_t\left(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}\right)$, then $\boldsymbol{P}_t$ equals:

$$\begin{aligned}
\boldsymbol{P}_t &= cov\left(\left(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}\right) - \boldsymbol{K}_t\left[h(\boldsymbol{x}_t) - h\left(\hat{\boldsymbol{x}}_{t|t-1}\right)\right]\right) + \boldsymbol{K}_t\boldsymbol{R}_{UKF}{\boldsymbol{K}_t}^T \\
&= cov\left((\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)\left(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1}\right)\right) + \boldsymbol{K}_t\boldsymbol{R}_{UKF}{\boldsymbol{K}_t}^T \\
&= (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)cov(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_{t|t-1})(\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)^T + \boldsymbol{K}_t\boldsymbol{R}_{UKF}{\boldsymbol{K}_t}^T \\
&= (\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)\boldsymbol{P}_{t|t-1}(\boldsymbol{I} - \boldsymbol{K}_t\boldsymbol{H}_t)^T + \boldsymbol{K}_t\boldsymbol{R}_{UKF}{\boldsymbol{K}_t}^T \\
&= \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{H}_t\boldsymbol{P}_{t|t-1} - \boldsymbol{P}_{t|t-1}{\boldsymbol{H}_t}^T{\boldsymbol{K}_t}^T \\
&\qquad + \boldsymbol{K}_t\left(\boldsymbol{H}_t\boldsymbol{P}_{t|t-1}{\boldsymbol{H}_t}^T + \boldsymbol{R}_{UKF}\right){\boldsymbol{K}_t}^T \\
&= \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{H}_t\boldsymbol{P}_{t|t-1} - \boldsymbol{P}_{t|t-1}{\boldsymbol{H}_t}^T{\boldsymbol{K}_t}^T + \boldsymbol{K}_t\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}{\boldsymbol{K}_t}^T
\end{aligned} \tag{5.54}$$

doi:10.6342/NTU202202468

Recalling the original concept of Kalman filter, minimizing the trace of estimated state covariance $\min_{\boldsymbol{K}_t} tr(\boldsymbol{P}_t)$, the optimal Kalman gain for the standard Kalman filter is formed by:

$$\frac{\partial\, tr(\boldsymbol{P}_t)}{\partial \boldsymbol{K}_t} = -2\left(\boldsymbol{H}_t \boldsymbol{P}_{t|t-1}\right)^T + 2\boldsymbol{K}_t \boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1} = 0 \tag{5.55}$$

$$\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t{}^T \boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}{}^{-1} \tag{5.56}$$

Additionally, with the optimal Kalman gain $\boldsymbol{K}_t$, the following equation is formed:

$$\boldsymbol{K}_t \boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}\boldsymbol{K}_t{}^T = \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t{}^T \boldsymbol{K}_t{}^T = \boldsymbol{K}_t \boldsymbol{H}_t \boldsymbol{P}_{t|t-1} \tag{5.57}$$

Therefore, the state covariance update function for standard Kalman filter and UKF is:

$$\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t \boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}\boldsymbol{K}_t{}^T \tag{5.58}$$

However, for the OCR-UKF, the Kalman gain $\boldsymbol{K}_{OCR,t}$ is designed to reject the effect of the outliers rather than minimize the trace of the state covariance. Thus, its state covariance should be updated by Equation (5.54) as:

$$\begin{aligned}\boldsymbol{P}_t = \boldsymbol{P}_{t|t-1} - \boldsymbol{K}_{OCR,t}\boldsymbol{H}_t \boldsymbol{P}_{t|t-1} - \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t{}^T \boldsymbol{K}_{OCR,t}{}^T \\ + \boldsymbol{K}_{OCR,t}\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}\boldsymbol{K}_{OCR,t}{}^T\end{aligned} \tag{5.59}$$

To calculate $\boldsymbol{H}_t$, we can utilize the optimal Kalman gain $\boldsymbol{K}_t$ for UKF calculated in Equation (3.29) and the derived relationship for optimal Kalman gain $\boldsymbol{K}_t$ and $\boldsymbol{H}_t$ in Equation (5.57) as:

$$\begin{cases}\boldsymbol{K}_t = \boldsymbol{P}_{\hat{\boldsymbol{x}}\hat{\boldsymbol{z}},t|t-1}\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}{}^{-1} \\ \boldsymbol{K}_t \boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}\boldsymbol{K}_t{}^T = \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t{}^T \boldsymbol{K}_t{}^T\end{cases} \tag{5.60}$$

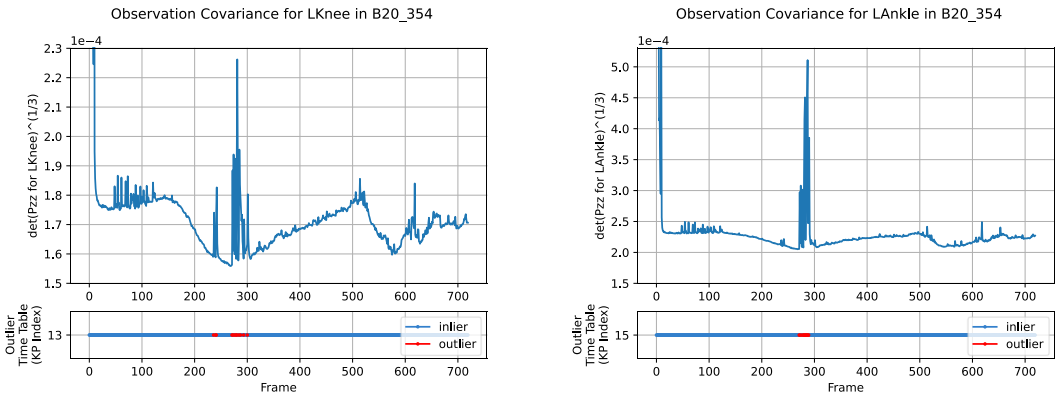
doi:10.6342/NTU202202468

Therefore, $\boldsymbol{H}_t$ can be calculated by:

$$\left(\boldsymbol{P}_{\hat{\boldsymbol{x}}\hat{\boldsymbol{z}},t|t-1}\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}{}^{-1}\right)\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1} = \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t{}^{T} \tag{5.61}$$

$$\begin{aligned}\boldsymbol{H}_t &= \left(\boldsymbol{P}_{t|t-1}{}^{-1}\boldsymbol{P}_{\hat{\boldsymbol{x}}\hat{\boldsymbol{z}},t|t-1}\right)^{T} \\ &= \boldsymbol{P}_{\hat{\boldsymbol{x}}\hat{\boldsymbol{z}},t|t-1}{}^{T}\,\boldsymbol{P}_{t|t-1}{}^{-1} \in \mathbb{R}^{m\times n}\end{aligned} \tag{5.62}$$

Imaginably, because the OCR Kalman gain $\boldsymbol{K}_{OCR,t}$ is not designed to minimize the state covariance, the state and observation covariance would be widened with the time update when the outlier components are rejected, as shown in Figure 5.15.



(a) (b)

Figure 5.15: Observation covariance variation while encountering the outliers
(a) observation covariance variation for LKnee in B20_354
(b) observation covariance variation for LAnkle in B20_354

However, it makes a good property that the OCR-UKF can catch the inliers even when the inliers are far from the predicted state after a long outlier interval as shown in Figure 5.16. Since the observation covariance is widened with time, the confidence score $s_l = e^{-k_{out}\frac{|\hat{\varepsilon}_l-\varepsilon_l|^2}{p_{l,l}}}$ would increase as $p_{l,l}$ is a diagonal element in observation covariance. Hence, the inlier region would also be expanded while the inlier threshold is fixed at $s_l = 0.5$.

doi:10.6342/NTU202202468

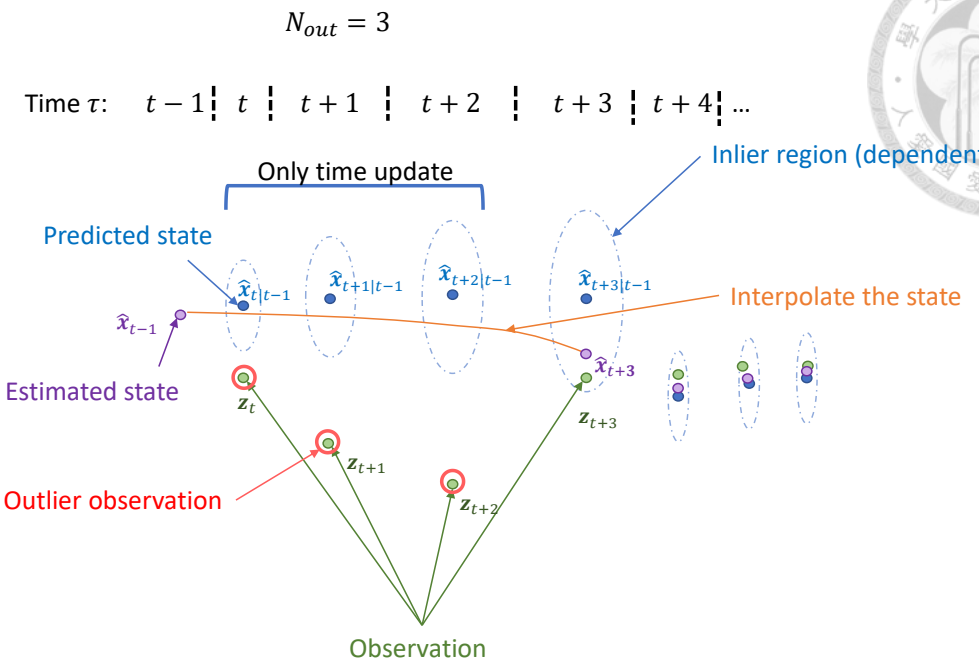


Figure 5.16: The concept of outlier rejection and inlier interpolation for OCR-UKF

Besides rejecting the outliers, it's also important to trace back the influence of the observations during the outlier interval once the inlier observations are caught, as shown in Figure 5.16. Since the outlier component rejection mechanism rejects all the effects of the observation in the outlier interval, the predicted state may be hugely biased without the new information from the observation. The measurement update only updates the estimated state once the inlier is caught at $\tau_{end} = t + N_{out}$ where N_{out} is the duration of the outlier interval. To trace the corresponding innovation during the outlier interval, we calculate the influence of innovation for the l-th component of $\boldsymbol{\varepsilon}_{\tau_{end}}$ on state as:

$$\boldsymbol{\epsilon}_{l,\tau_{end}} = \boldsymbol{k}_{l,\tau_{end}} \varepsilon_{l,\tau_{end}} \tag{5.63}$$

The vector $\boldsymbol{\epsilon}_{l,\tau_{end}}$ can represent the state change after the outlier interval for the l-th component.

doi:10.6342/NTU202202468

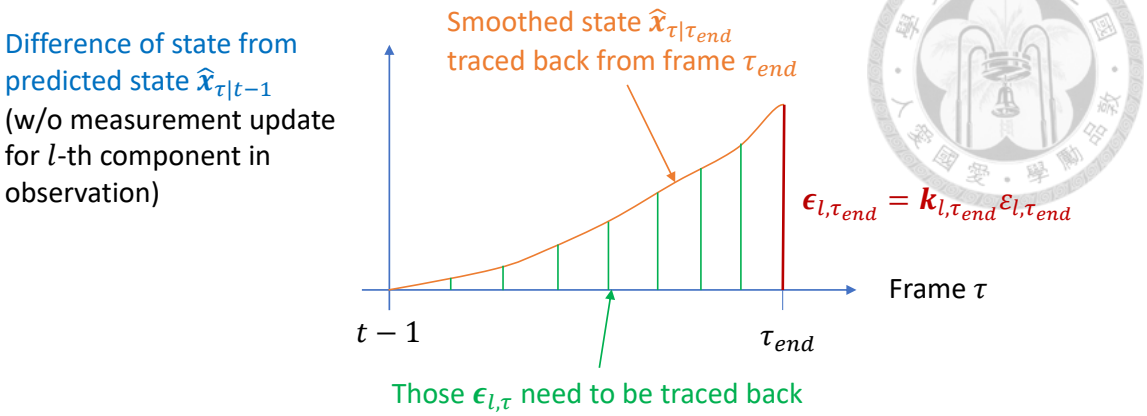


Figure 5.17: Illustration of tracing back state change (inlier interpolation)

To trace back the state change $\boldsymbol{\epsilon}_{l,\tau}$ for every frame in the outlier interval while considering the system dynamic, a variation of LQR tracking is applied:

$$\min_{\boldsymbol{u}_\tau} \frac{1}{2}\left[\sum_{\tau=t-1}^{\tau_{end}-1} \boldsymbol{u}_\tau{}^T \boldsymbol{R}_{in} \boldsymbol{u}_\tau\right] + \frac{1}{2}\left(\hat{\boldsymbol{\epsilon}}_{l,\tau_{end}} - \boldsymbol{\epsilon}_{l,\tau_{end}}\right)^T \boldsymbol{Q}_{in,\tau_{end}} \left(\hat{\boldsymbol{\epsilon}}_{l,\tau_{end}} - \boldsymbol{\epsilon}_{l,\tau_{end}}\right) \tag{5.64}$$

$$\text{s.t. } \hat{\boldsymbol{\epsilon}}_{l,\tau+1} = \boldsymbol{A}\hat{\boldsymbol{\epsilon}}_{l,\tau} + \boldsymbol{B}\boldsymbol{u}_\tau,\ \tau = t-1, t, \dots, \tau_{end} \text{ with } \hat{\boldsymbol{\epsilon}}_{l,t-1} = 0$$

The variation of LQR tracking only tracks the state change at the end and minimizes the weighting squared sum of the system input during the interval. Therefore, the state change during the outlier interval can be traced with the smoothest trajectory. Also, as the state transition function $f(\cdot)$ is linear the state change can be compensated with addition directly once the outlier interval ends for the l-th component:

$$\hat{\boldsymbol{x}}_\tau = \hat{\boldsymbol{x}}_{\tau|t-1} + \sum_{l=0}^{m} \hat{\boldsymbol{\epsilon}}_{l,\tau} \tag{5.65}$$

For the frames not including any outliers or whose outlier intervals haven't finished yet, their $\hat{\boldsymbol{\epsilon}}_{l,\tau}$ will be set as 0. After all, the procedures of the OCR-UKF can be illustrated in Figure 5.18.

doi:10.6342/NTU202202468

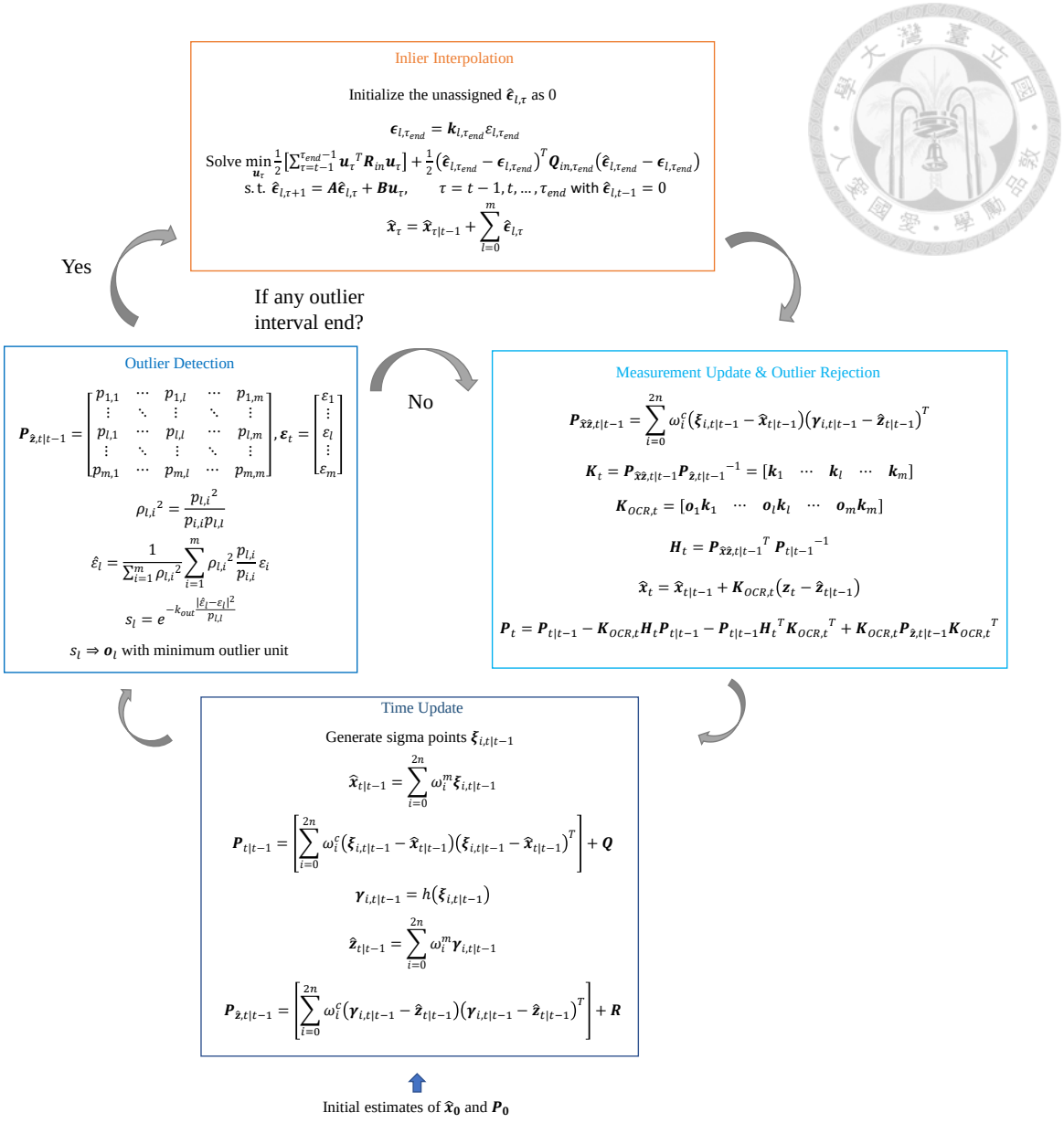


Figure 5.18: Illustration for the processes of OCR-UKF

doi:10.6342/NTU202202468

5.3.3 KKT condition and joint damper force for joint velocity

Besides the OCR-UKF, there are some other designs to improve the estimation performance in this work. They are KKT condition for joint velocity and the joint damper force. The two mechanisms are used to restrict the magnitudes of the joint velocity which are often huge for the transitional state with poor state initialization.

The KKT condition for joint velocity is to apply the well-known KKT conditions to restrict the state with lower bound and upper bound by adjusting the Kalman gain before determining the OCR Kalman gain $\boldsymbol{K}_{OCR,t}$. The detailed derivative will be shown in Appendix B.

In conclusion, the Kalman gain with the KKT condition is calculated by Equation (5.66) with the given state lower bound $\boldsymbol{x}_{lower}$ and state upper bound $\boldsymbol{x}_{upper}$.

$$\boldsymbol{K}_{KKT,t} = \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}^{-1} + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\boldsymbol{\varepsilon}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}^{-1} \tag{5.66}$$

The $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$ are determined with **Algorithm 5.2**

Algorithm 5.2: determine the Lagrange multiplier $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$
Input: state bounds $\boldsymbol{x}_{lower}$, $\boldsymbol{x}_{upper}$, innovation vector $\boldsymbol{\varepsilon}_t$ in the current frame, and other predicted information **Output**: the value of Lagrange multiplier $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_2$
Note: n is dimension of the state $\boldsymbol{x}_t$, $a_{t,i}$ is the i-th element of $\boldsymbol{a}_t$, and same for $b_{t,i}$, $\mu_{1,i}$, $\mu_{2,i}$

$\boldsymbol{a}_t \leftarrow \boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t\|t-1} - \boldsymbol{P}_{t\|t-1}\boldsymbol{H}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t\|t-1}^{-1}\boldsymbol{\varepsilon}_t$
$\boldsymbol{b}_t \leftarrow \hat{\boldsymbol{x}}_{t\|t-1} - \boldsymbol{x}_{upper} + \boldsymbol{P}_{t\|t-1}\boldsymbol{H}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t\|t-1}^{-1}\boldsymbol{\varepsilon}_t$
$c_t \leftarrow \boldsymbol{\varepsilon}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t\|t-1}^{-1}\boldsymbol{\varepsilon}_t$
for $i \leftarrow 1,\ldots,n$ **do**
  if $a_{t,i} \geq 0$ **do**
  $\mu_{1,i} \leftarrow \frac{1}{c_t}a_{t,i}$

doi:10.6342/NTU202202468

```
        μ_{2,i} ← 0
    else if a_{t,i} ≤ 0 and b_{t,i} ≤ 0 do
        μ_{1,i} ← 0
        μ_{2,i} ← 0
    else do
        μ_{1,i} ← 0
        μ_{2,i} ← (1/c_t) b_{t,i}
    if end
return μ_1, μ_2
```

With it, the state can be bounded in the interval between $\boldsymbol{x}_{lower}$ and $\boldsymbol{x}_{upper}$.

However, the joint velocities are not always in the low magnitude intervals. Especially for intense exercise, the joint velocity may raise dramatically for a short time, then, drop down suddenly. Therefore, the state bounds are only used for the unhuman motion with a relatively wide restricted state interval.

For real human motion, it's hard to maintain a high joint velocity for a long time. Besides the stopping consciousness of humans, we think that there are some other impedances coming from the human body. We regard the impedance as the damper force to generate the resistive force when the joint velocity is large. Thus, the state transition function is rewritten as:

$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}) = \boldsymbol{A}_{UKF}\boldsymbol{x}_{t-1} + \boldsymbol{B}_{UKF}\boldsymbol{u}_{damp,t-1} \tag{5.67}$$

$$\boldsymbol{u}_{damp,t-1} = c_{damp} \cdot \begin{bmatrix} \boldsymbol{\Theta}_{\Delta,t-1}^T & \boldsymbol{T}_{\Delta,t-1}^T \end{bmatrix}^T \tag{5.68}$$

The joint velocity and head-world transformation velocity come from $\boldsymbol{x}_{t-1}$:

$$\boldsymbol{x}_{t-1} = \begin{bmatrix} \boldsymbol{\Theta}_{t-1}^T & \boldsymbol{\Theta}_{\Delta,t-1}^T & \boldsymbol{T}_{t-1}^T & \boldsymbol{T}_{\Delta,t-1}^T \end{bmatrix}^T \tag{5.69}$$

doi:10.6342/NTU202202468

After the OCR-UKF and the velocity constraints, the estimated trajectory is shown in Figure 5.19. Compared with the RawSK, most of the outliers are rejected successfully. However, the trajectory still looks rough and shaky, which makes it unlike a perfect estimation result. To handle this problem, an iterative LQR motion smoother will be proposed in Section 5.5.

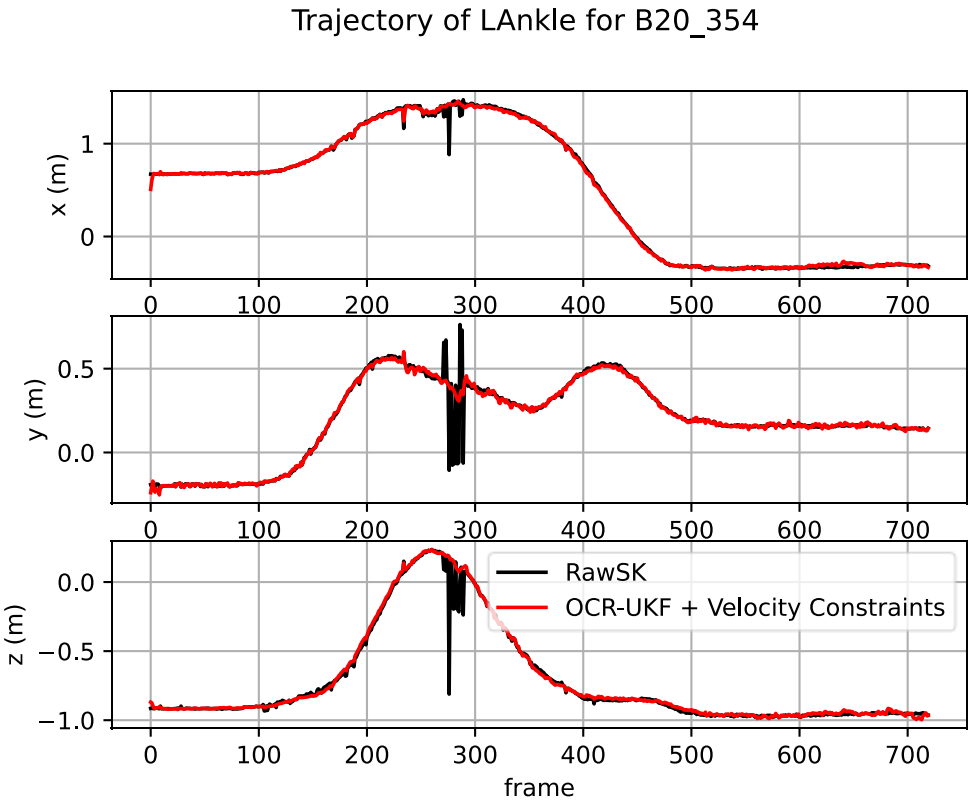


Figure 5.19: The estimated trajectory of LAnkle for B20_354 with OCR-UKF

doi:10.6342/NTU202202468

5.4 Initial Inversed Kinematic Estimation

As mentioned in Section 5.3, the proposed OCR-UKF needs a guess initial state $\hat{\boldsymbol{x}}_0$ in the first frame before the filtering process. With the converge property of Kalman filter, the initial guess for the state is able to accept some tolerance. However, if the guessed initial state has too large errors so that the estimated state is out of the convergence region, the filter process may work improperly. Thus, a good estimation of the initial state is important. The proposed procedure here is similar to the solving of **inversed kinematic** of the human skeleton with **the RawSK in the first frame**.

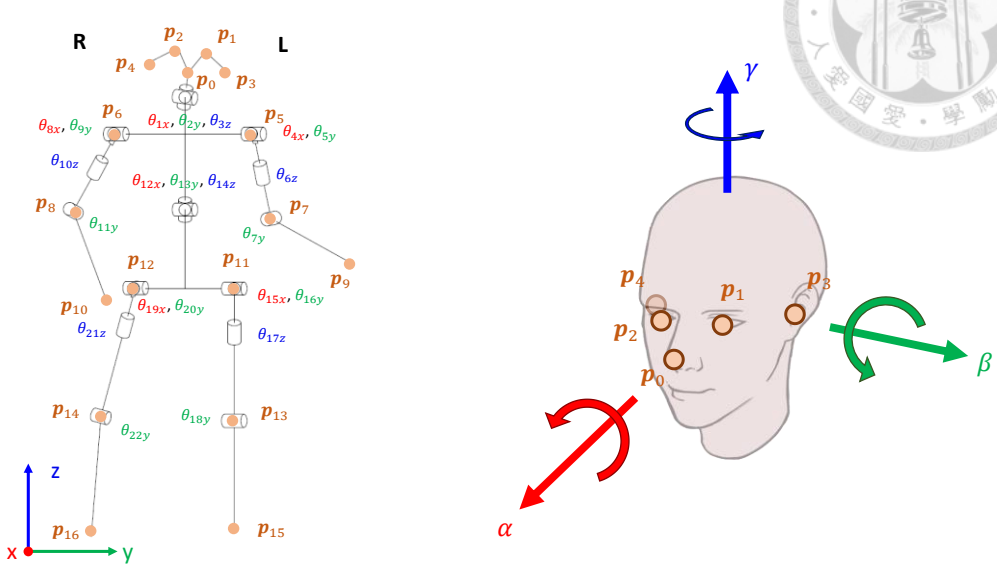
The initial parameters that need to be estimated are:

$$\boldsymbol{x}_{pos,init} = [\boldsymbol{\Theta}_{init}^T \quad \boldsymbol{T}_{init}^T]^T \in \mathbb{R}^{28} \tag{5.70}$$

$$\begin{cases} \boldsymbol{\Theta}_{init} = [\theta_{1,init} \quad \theta_{2,init} \quad \cdots \quad \theta_{22,init}]^T \\ \boldsymbol{T}_{init} = [p_{0,x,init} \quad p_{0,y,init} \quad p_{0,z,init} \quad \alpha_{init} \quad \beta_{init} \quad \gamma_{init}]^T \end{cases} \tag{5.71}$$

doi:10.6342/NTU202202468

These parameters are used as the state describing human pose as Figure 5.20.



(a) (b)

Figure 5.20: Illustration for the initial estimated states
(a) the initial pose and skeleton joint angle $\boldsymbol{\theta}$ (b) the initial head pose and head-world transformation parameters $\boldsymbol{T}$

The estimated parameters can be divided into five parts: 1) **Head coordinate estimation** ($\boldsymbol{T}_{init}$), 2) **Shoulder coordinate estimation** ($\theta_{1,init}, \theta_{2,init}, \theta_{3,init}$), 3) **Hip coordinate estimation** ($\theta_{12,init}, \theta_{13,init}, \theta_{14,init}$), 4) **Elbow and knee angle estimation** ($\theta_{7,init}, \theta_{11,init}, \theta_{18,init}, \theta_{22,init}$), and 5) **Shoulder and hip joint angle estimation** ($\theta_{4,init}, \theta_{5,init}, \theta_{6,init}, \theta_{8,init}, \theta_{9,init}, \theta_{10,init}, \theta_{15,init}, \theta_{16,init}, \theta_{17,init}, \theta_{19,init}, \theta_{20,init}, \theta_{21,init}$).

doi:10.6342/NTU202202468

1). **Head coordinate estimation** ($\boldsymbol{T}_{init}$)

As mentioned in Section 5.1, the forward kinematic of the proposed human kinematic model starts from the head. Therefore, the first step is to estimate the head-world transformation parameters $\boldsymbol{T}_{init}$ in Figure 5.20 (b).

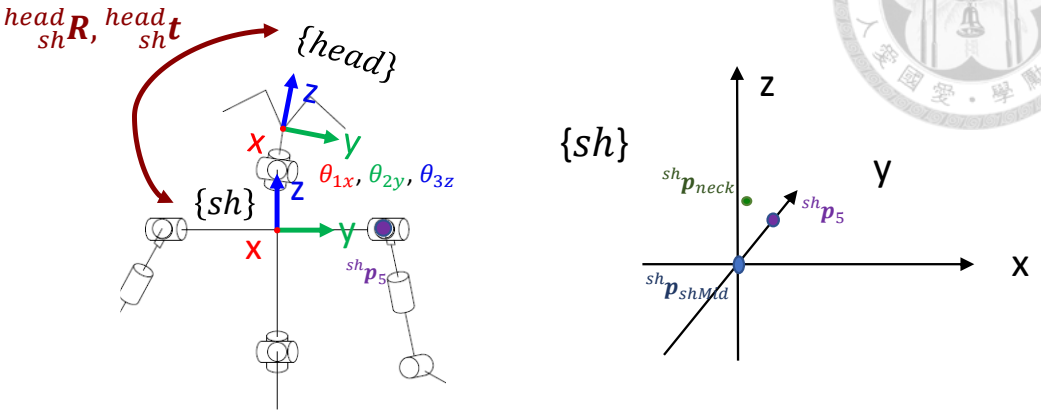
The first three elements of $\boldsymbol{T}_{init}$, $[p_{0,x,init} \quad p_{0,y,init} \quad p_{0,z,init}]^T$, are estimated as the position of the nose keypoint $\boldsymbol{p}_0$ in the first frame directly. To estimate the roll-pitch-yaw angles of the transformation between world coordinate and head coordinate, the head keypoint positions in $\{head\}$ and the positions of the head keypoints in $\{world\}$ are utilized.

Since there are 5 points in the head, the 3D transformation matrix estimation method presented in Section 3.6 can be used to find the relative transformation ${}^{world}_{head}\boldsymbol{T}$. Taking advantage of the rotation matrix ${}^{world}_{head}\boldsymbol{R}$ in ${}^{world}_{head}\boldsymbol{T}$, the roll-pitch-yaw angles α_{init}, β_{init}, γ_{init} can be obtained with the ZYX rotation matrix decomposition method mentioned in **Algorithm 3.1**.

$$\alpha_{init}, \beta_{init}, \gamma_{init} \leftarrow ZYX\_Rotation\_Decomposition\left({}^{world}_{head}\boldsymbol{R}\right) \tag{5.72}$$

doi:10.6342/NTU202202468

2). **Shoulder coordinate estimation** $(\theta_{1,init}, \theta_{2,init}, \theta_{3,init})$



(a) (b)

Figure 5.21: Illustration for head-shoulder transformation
(a) head-shoulder transformation (b) shoulder coordinate definition

To estimate the initial values of $\theta_{1,init}, \theta_{2,init}, \theta_{3,init}$, the rotation matrix $^{head}_{sh}\boldsymbol{R}$ from head coordinate $\{head\}$ to shoulder coordinate $\{sh\}$ should be estimated first. The head coordinate $\{head\}$ for the initial frame has been estimated in the 1) **Head coordinate estimation** ($\boldsymbol{T}_{init}$). For the shoulder coordinate $\{sh\}$, its definition is shown in Figure 5.21(b). The middle point of the shoulder points $\boldsymbol{p}_{shMid} = {(\boldsymbol{p}_5 + \boldsymbol{p}_6)}/{2}$ is set as the origin of $\{sh\}$. The left shoulder keypoint $\boldsymbol{p}_5$ is set lying on the positive y-axis of $\{sh\}$. Then, the neck rotation center $\boldsymbol{p}_{neck}$, calculated with the spine parameter $^{head}\boldsymbol{p}_{neck}$ and world-head transformation $^{world}_{head}\boldsymbol{T}$, is set lying on the YZ-plane of $\{sh\}$ with positive z-value. With these three conditions, the shoulder frame $\{sh\}$ and its transformation matrix $^{world}_{sh}\boldsymbol{T}$ from $\{world\}$ can estimated easily. To obtain the rotation matrix $^{head}_{sh}\boldsymbol{R}$ affected by $\theta_{1,init}, \theta_{2,init}, \theta_{3,init}$, Equation (5.73)-(5.74) are applied:

doi:10.6342/NTU202202468

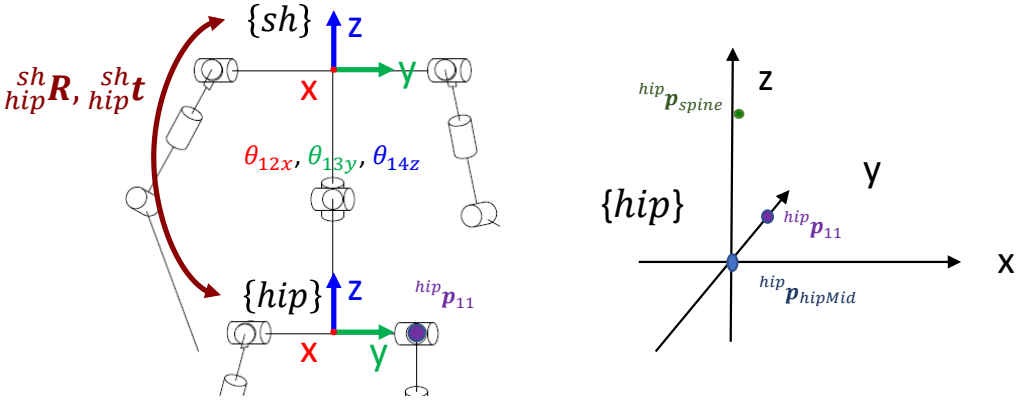
$$ {}^{head}_{sh}\boldsymbol{T} = {}^{world}_{head}\boldsymbol{T}^{-1} \cdot {}^{world}_{sh}\boldsymbol{T} \tag{5.73} $$

$$ \begin{bmatrix} {}^{head}_{sh}\boldsymbol{R} & {}^{head}_{sh}\boldsymbol{t} \\ \boldsymbol{0}_{3\times1} & 1 \end{bmatrix} = {}^{head}_{sh}\boldsymbol{T} \tag{5.74} $$

Thus, the $\theta_{1,init}, \theta_{2,init}, \theta_{3,init}$ can be estimated as:

$$ \theta_{1,init}, \theta_{2,init}, \theta_{3,init} \leftarrow ZYX\_Rotation\_Decomposition\left({}^{head}_{sh}\boldsymbol{R}\right) \tag{5.75} $$

3). **Hip coordinate estimation** $(\theta_{12,init}, \theta_{13,init}, \theta_{14,init})$



(a) (b)

Figure 5.22: Illustration for shoulder-hip transformation
(a) shoulder-hip transformation (b) hip coordinate definition

Similar to the shoulder coordinate $\{sh\}$, the hip coordinate $\{hip\}$ is defined with three conditions as Figure 5.25(b) shows. The middle point of the hip points $\boldsymbol{p}_{hipMid} = (\boldsymbol{p}_{11} + \boldsymbol{p}_{12})/_2$ is set as the origin of $\{hip\}$. The left hip keypoint $\boldsymbol{p}_{11}$ is set lying on the positive y-axis of $\{hip\}$. Then, the spine rotation center $\boldsymbol{p}_{spine}$, calculated with the spine parameter ${}^{sh}\boldsymbol{p}_{spine}$ and world-shoulder transformation ${}^{world}_{sh}\boldsymbol{T}$, is set lying on the YZ-plane of $\{hip\}$ with a positive z-value. Utilized the three conditions, the transformation matrix ${}^{world}_{hip}\boldsymbol{T}$ can be calculated.

doi:10.6342/NTU202202468

Since the relative rotation ${}^{sh}_{hip}\boldsymbol{R}$ from $\{sh\}$ to $\{hip\}$ is only influenced by $\theta_{12}, \theta_{13}, \theta_{14}$, $\theta_{12,init}, \theta_{13,init}, \theta_{14,init}$ can be estimated with ${}^{sh}_{hip}\boldsymbol{R}$ directly. Applying the same method in 2) **Shoulder coordinate estimation**:

$$ {}^{sh}_{hip}\boldsymbol{T} = {}^{world}_{sh}\boldsymbol{T}^{-1} \cdot {}^{world}_{hip}\boldsymbol{T} \tag{5.76} $$

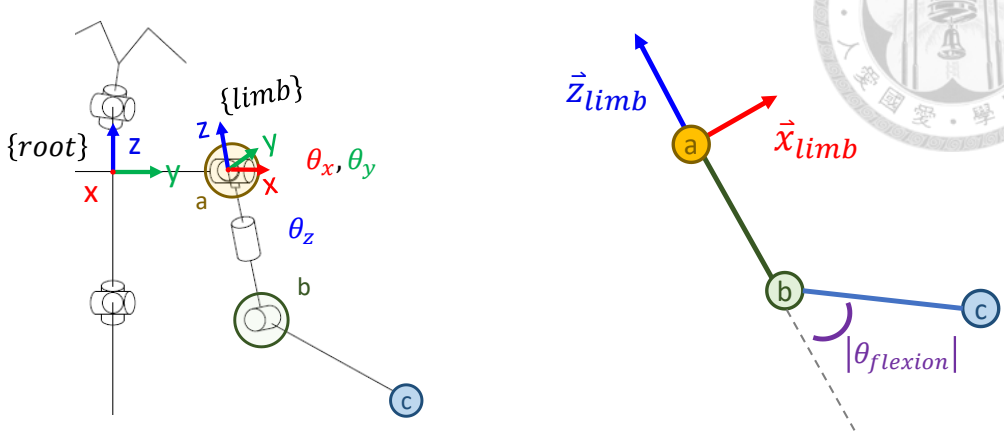
$$ \begin{bmatrix} {}^{sh}_{hip}\boldsymbol{R} & {}^{sh}_{hip}\boldsymbol{t} \\ \boldsymbol{0}_{3\times1} & 1 \end{bmatrix} = {}^{sh}_{hip}\boldsymbol{T} \tag{5.77} $$

Thus, the $\theta_{12,init}, \theta_{13,init}, \theta_{14,init}$ can be estimated as:

$$ \theta_{12,init}, \theta_{13,init}, \theta_{14,init} \leftarrow ZYX\_Rotation\_Decomposition\left({}^{sh}_{hip}\boldsymbol{R}\right) \tag{5.78} $$

doi:10.6342/NTU202202468

4). **Elbow and knee angle estimation** $(\theta_{7,init}, \theta_{11,init}, \theta_{18,init}, \theta_{22,init})$



(a) (b)

Figure 5.23: Illustration for limb joint angle estimation
(a) limb coordinate $\{limb\}$ and its parent coordinate $\{root\}$ (b) XY-plane definition of $\{limb\}$ and the flexion angle

After part 1) to part 3), the initial states affecting the main body pose have been estimated. The remaining states influence the poses of each limb. For the four limbs in a human skeleton, their state initialization methods are the same.

As Figure 5.23(a) shows, from the keypoint close to the main body to the endpoint of the limb, the three keypoints are denoted as "a (shoulder or hip)", "b (elbow or knee)", and "c (wrist or ankle)". In this part, the initial elbow and knee angles $(\theta_{7,init}, \theta_{11,init}, \theta_{18,init}, \theta_{22,init})$, which are the flexion angle $\theta_{flexion}$ shown in Figure 5.23(b), will be estimated.

As shown in Figure 5.23(b), the magnitude of the flexion angle $|\theta_{flexion}|$ can be calculated with the vector $\overrightarrow{ab}$ from "a" to "b" and the vector $\overrightarrow{bc}$ from "b" to "c" as:

$$|\theta_{flexion}| = \pi - \arctan2(\|\overrightarrow{ab} \times \overrightarrow{bc}\|, |\overrightarrow{ab} \cdot \overrightarrow{bc}|) \quad (5.79)$$

doi:10.6342/NTU202202468

The angle $\arctan2\left(\left\|\overrightarrow{ab}\times\overrightarrow{bc}\right\|,\left|\overrightarrow{ab}\cdot\overrightarrow{bc}\right|\right)$ is the included angle between $\overrightarrow{ab}$ and $\overrightarrow{bc}$. Since $\left\|\overrightarrow{ab}\times\overrightarrow{bc}\right\|\geq 0$, the included angle must be smaller than π.

After acquiring the magnitude of the flexion angle $\left|\theta_{flexion}\right|$, the sign of $\theta_{flexion}$ is dependent on what the angle is. Referring to [84: DSHS.WA 2022] and the proposed human skeleton kinematic model in Section 5.1, the flexion angles of elbows should always be negative, and the ones of knees should always be positive, i.e.:

$$\begin{cases}\theta_{7,init}=-\left|\theta_{7,init}\right|\\ \theta_{11,init}=-\left|\theta_{11,init}\right|\end{cases} \tag{5.80}$$

$$\begin{cases}\theta_{18,init}=\left|\theta_{18,init}\right|\\ \theta_{22,init}=\left|\theta_{22,init}\right|\end{cases} \tag{5.81}$$

5). **Shoulder and hip joint angle estimation**

$(\theta_{4,init},\theta_{5,init},\theta_{6,init},\theta_{8,init},\theta_{9,init},\theta_{10,init},\theta_{15,init},\theta_{16,init},\theta_{17,init},\theta_{19,init},\theta_{20,init},\theta_{21,init})$

Continuing the denotation in part 4) **Elbow and knee angle estimation**, subsequently, we defined two coordinates $\{root\}$ and $\{limb\}$ as shown in Figure 5.23(a). $\{root\}$ is the parent coordinate of the limb, $\{sh\}$ for the arms, $\{hip\}$ for the legs. $\{limb\}$ is the coordinate of the limb, which is defined with the following conditions as shown in Figure 5.23(b). The origin of $\{limb\}$ is defined at the point "a". The z-axis of $\{limb\}$ is parallel and has the same direction as the vector $\overrightarrow{ba}$ from "b" to "a". While the points "a", "b" and "c" are not colinear, the point "c" is set lying on the XZ-plane of $\{limb\}$ so that there are two possible unit x-vector $\vec{x}_{limb}$ of $\{limb\}$. To make the

doi:10.6342/NTU202202468

solution unique, another constrain is that the angle between the unit x-vector $\vec{\boldsymbol{x}}_{limb}$ of $\{limb\}$ and the unit x-vector $\vec{\boldsymbol{x}}_{root}$ of $\{root\}$ should be equal or smaller than $\pi/2$, i.e.:

$$\vec{\boldsymbol{x}}_{limb}^{T} \cdot \vec{\boldsymbol{x}}_{root} > 0 \tag{5.82}$$

After that, the limb coordinate $\{limb\}$ is defined and ${}^{world}_{limb}\boldsymbol{T}$ can be obtained.

To estimate the corresponding joint angles $\theta_x, \theta_y, \theta_z$ of the limb, the rotation matrix ${}^{root}_{limb}\boldsymbol{R}$ from $\{root\}$ to $\{limb\}$ is received with:

$${}^{root}_{limb}\boldsymbol{T} = {}^{world}_{root}\boldsymbol{T}^{-1} \cdot {}^{world}_{limb}\boldsymbol{T} \tag{5.83}$$

$$\begin{bmatrix} {}^{root}_{limb}\boldsymbol{R} & {}^{root}_{limb}\boldsymbol{t} \\ \boldsymbol{0}_{3\times1} & 1 \end{bmatrix} = {}^{root}_{limb}\boldsymbol{T} \tag{5.84}$$

With the kinematic structure in Figure 5.23(a), we can know the rotation matrix ${}^{root}_{limb}\boldsymbol{R}$ can be represented as:

$${}^{root}_{limb}\boldsymbol{R} = \boldsymbol{R}_y(\theta_y)\boldsymbol{R}_x(\theta_x)\boldsymbol{R}_z(\theta_z) \tag{5.85}$$

To acquire the rotation angles, the YXZ rotation decomposition described in **Algorithm 3.2** would be applied:

$$\theta_x, \theta_y, \theta_z \leftarrow YXZ\_Rotation\_Decomposition({}^{root}_{limb}\boldsymbol{R}) \tag{5.86}$$

The rotation angles $\theta_x, \theta_y, \theta_z$ correspond to the rotation angles in the shoulders and hips as Table 5.2.

Table 5.2

Rotation joint angle correspondences for limbs

Joint Angle in a Limb	Joint Angle in Human Skeleton
θ_x	$\theta_{4,init}, \theta_{8,init}, \theta_{15,init}, \theta_{19,init}$
θ_y	$\theta_{5,init}, \theta_{9,init}, \theta_{16,init}, \theta_{20,init}$
θ_z	$\theta_{6,init}, \theta_{10,init}, \theta_{17,init}, \theta_{21,init}$

doi:10.6342/NTU202202468

As mentioned in part 5) **Shoulder and hip joint angle estimation**, the initialization only works when the points of the limbs are not colinear. While the keypoints are colinear for one of the limbs, the initialization would output the initial state including NaNs (not a number). The situation would also happen when one of the limb keypoints is missed or there are more than one head keypoints missed. To reduce the probability of failure and also increase its stability, the state initialization would be run for the first 10 frames and average the estimated states excluding the NaNs.

doi:10.6342/NTU202202468

5.5 Iterative LQR Motion Smoother

After the OCR-UKF filtering, we can see there are still some small noise in the joint states so that the filtered skeleton looks shaky. The reason comes from that the Kalman filter is a stochastic filter. As the assumption in Equation (5.36)-(5.37), there are process noises when the state is updating. It's reasonable that the filtered results are shaky.

However, for human motion estimation, not only the position but also the higher order terms, like velocity and acceleration, should be estimated accurately. The shaky motion of the skeleton would lead to huge errors in the discrete derivation of the positions. To reduce the noises while considering the system dynamic, i.e. constant velocity model in Equation (5.43), the linear quadratic tracking (LQR tracking) mentioned in Section 3.8 is applied.

The denoise smoothing process can be formed as an optimization problem as:

$$\min_{\boldsymbol{u}_t} \frac{1}{2}\left[\sum_{t=0}^{N-1}(\boldsymbol{x}_t-\hat{\boldsymbol{x}}_t)^T\boldsymbol{Q}_{lqr1}(\boldsymbol{x}_t-\hat{\boldsymbol{x}}_t)+\boldsymbol{u}_t{}^T\boldsymbol{R}_{lqr1}\boldsymbol{u}_t\right] + \frac{1}{2}(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N)^T\boldsymbol{Q}_{lqr1,N}(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N) \quad (5.87)$$

$$\text{s.t. } \boldsymbol{x}_t=\boldsymbol{A}_{UKF}\boldsymbol{x}_{t-1}+\boldsymbol{B}_{UKF}\boldsymbol{u}_t,\ t=0,1,\dots,N-1 \text{ given } \boldsymbol{x}_0$$

The reference trajectory $\hat{\boldsymbol{x}}_t$ to track is the output of the OCR-UKF, and $\boldsymbol{Q}_{lqr1}$, $\boldsymbol{R}_{lqr1}$, $\boldsymbol{Q}_{lqr1,N}$ are the handcrafted weights set as $\boldsymbol{Q}_{lqr1}=\boldsymbol{Q}_{lqr1,N}=\boldsymbol{I}_{56}$ and $\boldsymbol{R}_{lqr1}=\boldsymbol{10^3}\cdot\boldsymbol{I}_{28}$. The optimal solution of $\boldsymbol{x}_t$ introduced in Section 3.8 is the smoothed trajectory by the LQR tracking.

doi:10.6342/NTU202202468

The concept of using LQR tracking as a smoothing filter is that the denoise process is like a trade-off between the position accuracy and the smoothness of the estimated trajectory. When the smooth state dynamic is fit, the constant velocity model for example, noise term is supposed to come from the input signals. For LQR tracking, it's also a trade-off between the state tracking errors and the magnitude of input when the tracking input is zero. In the constant velocity model, the roughness of the trajectories causes by the additional acceleration $\boldsymbol{u}_t$. Using the LQR tracking as a smoothing filter can effectively reduce the roughness and keep the small tracking errors for state $\boldsymbol{x}_t$ while considering the given system dynamic.

After applying the smoothing technique, the overall roughness of the joint state seems to decrease while the overall trajectory is close to the output of OCR-UKF, as shown in Figure 5.24.

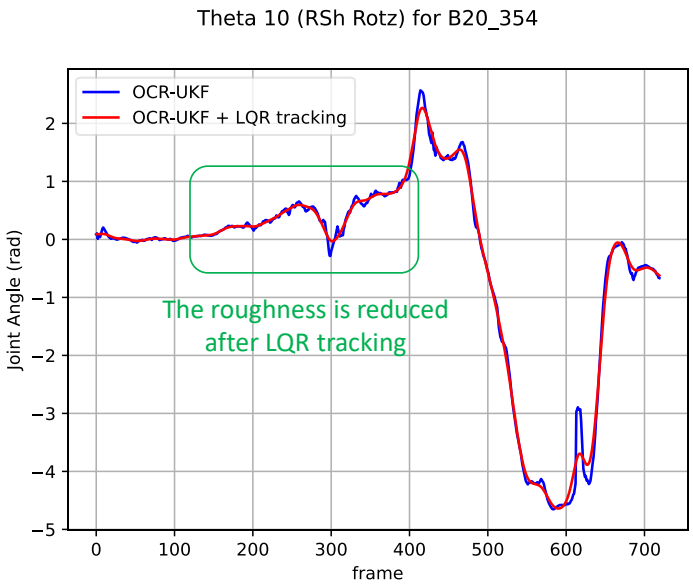


Figure 5.24: The comparison of the smoothness before and after the LQR tracking

doi:10.6342/NTU202202468

Nonetheless, it seems to cause small additional errors for the keypoint trajectory, as shown in Figure 5.25.

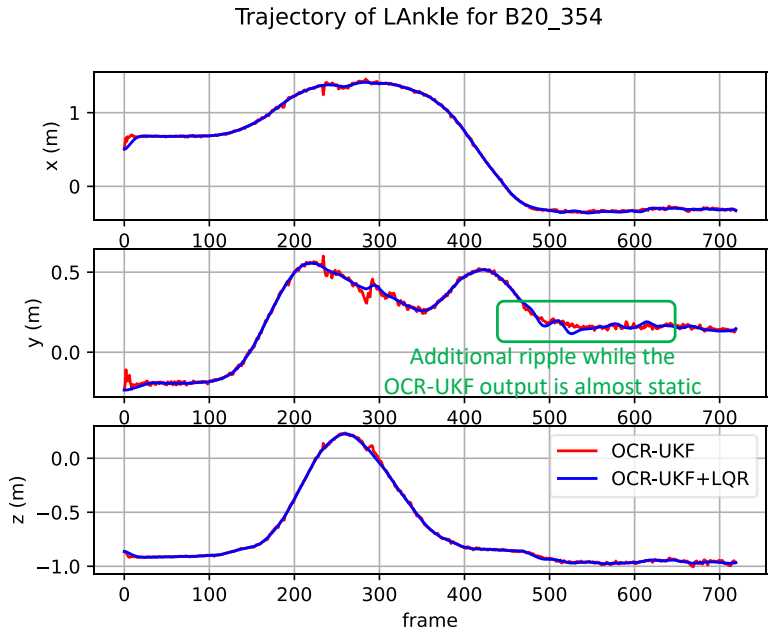


Figure 5.25: The additional error comes from the directly LQR tracking

The reason for these additional errors causes by the independent tracking for each joint angle. The accuracy of output of OCR-UKF is based on the forward kinematic function $h(\boldsymbol{x}_t)$ which takes the effect from all joint states to every keypoint position. In the LQR tracking, since the weights are set as $\boldsymbol{Q}_{lqr} = \boldsymbol{Q}_{lqr,N} = \boldsymbol{I}_{56}$ and $\boldsymbol{R}_{lqr} = \mathbf{10^3} \cdot \boldsymbol{I}_{28}$, every joint track their own trajectories independently. Due to this reason, the keypoint position calculated with the forward kinematic function may generate some additional errors because the coordinate among the joints is ignored. As the result, for the keypoints far from the head, the start of the body chain in the forward kinematic function, the additional error would be greater since there are more uncooperative joints affecting the positions of the keypoints.

doi:10.6342/NTU202202468

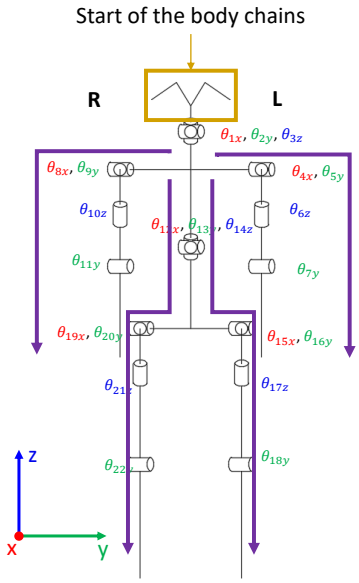


Figure 5.26: Body chains of the proposed kinematic model for human skeleton

To solve this issue, instead of the joint state $\boldsymbol{x}_t$, the target we need to track is the keypoint positions $\hat{\boldsymbol{z}}_t = h(\hat{\boldsymbol{x}}_t)$:

$$\min_{\boldsymbol{u}_t} \frac{1}{2}\left[\sum_{t=0}^{N-1}(h(\boldsymbol{x}_t)-\hat{\boldsymbol{z}}_t)^T\boldsymbol{Q}_{lqr2}(h(\boldsymbol{x}_t)-\hat{\boldsymbol{z}}_t)+\boldsymbol{u}_t{}^T\boldsymbol{R}_{lqr2}\boldsymbol{u}_t\right] + \frac{1}{2}(h(\boldsymbol{x}_N)-\hat{\boldsymbol{z}}_N)^T\boldsymbol{Q}_{lqr2,N}(h(\boldsymbol{x}_N)-\hat{\boldsymbol{z}}_N) \quad (5.88)$$

$$\text{s.t. } \boldsymbol{x}_t = \boldsymbol{A}_{UKF}\boldsymbol{x}_{t-1} + \boldsymbol{B}_{UKF}\boldsymbol{u}_t,\ t = 0,1,\dots,N-1 \text{ given } \boldsymbol{x}_0$$

However, after adjusting the chasing target as Equation (5.88), the optimization problem is turned into a nonlinear problem and takes a huge time to solve.

doi:10.6342/NTU202202468

To speed up the processing time with the analytic solution of LQR tracking, we use the linearization technique as:

$$\begin{aligned} J &= \frac{1}{2}\left[\sum_{t=0}^{N-1}(h(\boldsymbol{x}_t)-\hat{\boldsymbol{z}}_t)^T\boldsymbol{Q}_{lqr2}(h(\boldsymbol{x}_t)-\hat{\boldsymbol{z}}_t)+{\boldsymbol{u}_t}^T\boldsymbol{R}_{lqr2}\boldsymbol{u}_t\right] \\ &\quad+\frac{1}{2}(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N)^T\boldsymbol{Q}_{lqr2,N}(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N) \\ &\approx \frac{1}{2}\left[\sum_{t=0}^{N-1}(\boldsymbol{x}_t-\tilde{\boldsymbol{x}}_t)^T{\boldsymbol{C}_{\breve{\boldsymbol{x}}_t}}^T\boldsymbol{Q}_{lqr2}\boldsymbol{C}_{\breve{\boldsymbol{x}}_t}(\boldsymbol{x}_t-\tilde{\boldsymbol{x}}_t)+{\boldsymbol{u}_t}^T\boldsymbol{R}_{lqr2}\boldsymbol{u}_t\right] \\ &\quad+\frac{1}{2}(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N)^T{\boldsymbol{C}_{\breve{\boldsymbol{x}}_N}}^T\boldsymbol{Q}_{lqr2,N}\boldsymbol{C}_{\breve{\boldsymbol{x}}_N}(\boldsymbol{x}_N-\hat{\boldsymbol{x}}_N) \end{aligned} \tag{5.89}$$

$\breve{\boldsymbol{x}}_t$ is the tracked state generated in the previous iteration, $\boldsymbol{C}_{\breve{\boldsymbol{x}}_t}$ is the Jacobian matrix of $h(\boldsymbol{x}_t)$ at $\boldsymbol{x}_t=\breve{\boldsymbol{x}}_t$:

$$\boldsymbol{C}_{\breve{\boldsymbol{x}}_t}=\frac{\partial h}{\partial \boldsymbol{x}_t}|_{\boldsymbol{x}_t=\breve{\boldsymbol{x}}_t} \tag{5.90}$$

$\tilde{\boldsymbol{x}}_t$ is the new tracked state in this iteration calculated as Equation where $pinv(\boldsymbol{C}_{\breve{\boldsymbol{x}}_t})$ is the pseudo inversed matrix of $\boldsymbol{C}_{\breve{\boldsymbol{x}}_t}$:

$$\tilde{\boldsymbol{x}}_t=\breve{\boldsymbol{x}}_t+pinv(\boldsymbol{C}_{\breve{\boldsymbol{x}}_t})\cdot(\hat{\boldsymbol{z}}_t-h(\breve{\boldsymbol{x}}_t)) \tag{5.91}$$

Therefore, the linearized keypoint tracking problem can be formulated as:

$$\begin{aligned} \min_{\boldsymbol{u}_t}\ &\frac{1}{2}\left[\sum_{t=0}^{N-1}(\boldsymbol{x}_t-\tilde{\boldsymbol{x}}_t)^T\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_t}(\boldsymbol{x}_t-\tilde{\boldsymbol{x}}_t)+{\boldsymbol{u}_t}^T\boldsymbol{R}_{lqr2}\boldsymbol{u}_t\right] \\ &+\frac{1}{2}(\boldsymbol{x}_N-\tilde{\boldsymbol{x}}_N)^T\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_N}(\boldsymbol{x}_N-\tilde{\boldsymbol{x}}_N) \\ \text{s.t.}\ \ &\boldsymbol{x}_t=\boldsymbol{A}_{UKF}\boldsymbol{x}_{t-1}+\boldsymbol{B}_{UKF}\boldsymbol{u}_t,\ t=0,1,\dots,N-1\ \text{ given }\ \boldsymbol{x}_0 \end{aligned} \tag{5.92}$$

doi:10.6342/NTU202202468

The only difference to the classical LQR tracking is the $\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_t}$ is time-varying. Despite that, the analytic solution derived in Equation (3.86) still works for the time-varying weights.

However, since the Jacobian matrix of the forward kinematic function $\boldsymbol{C}_{\breve{\boldsymbol{x}}_t}$ is always zero for the velocity state, $\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_t} = \boldsymbol{C}_{\breve{\boldsymbol{x}}_t}{}^T \boldsymbol{Q}_{lqr2} \boldsymbol{C}_{\breve{\boldsymbol{x}}_t}$ has zero weighting for the velocity tracking. To make up for this issue, $\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_t}$ would be added with other weights:

$$\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_t} = \boldsymbol{C}_{\breve{\boldsymbol{x}}_t}{}^T \boldsymbol{Q}_{lqr2} \boldsymbol{C}_{\breve{\boldsymbol{x}}_t} + \boldsymbol{Q}_{lqr2,vel} \tag{5.93}$$

The $\boldsymbol{Q}_{lqr2,vel}$ is a diagonal weight for the velocity state as Equation (5.94) where G_{vel} is a constant gain for the velocity weights.

$$\boldsymbol{Q}_{lqr2,vel} = \begin{bmatrix} \boldsymbol{0}_{22\times22} & \boldsymbol{0}_{22\times22} & \boldsymbol{0}_{22\times6} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{22\times22} & G_{vel} \cdot \boldsymbol{I}_{22} & \boldsymbol{0}_{22\times6} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times6} & \boldsymbol{0}_{6\times6} \\ \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times6} & G_{vel} \cdot \boldsymbol{I}_{6} \end{bmatrix} \tag{5.94}$$

In addition, because the trajectory filtered after the OCR-UKF is shaky, the velocity state filtered after the direct LQR tracking would be more accurate. Therefore, the tracking target $\bar{\boldsymbol{x}}_t$ after adding the velocity weight would be calculated by Equation (5.95):

$$\bar{\boldsymbol{x}}_t = \begin{bmatrix} \widetilde{\boldsymbol{\Theta}}_t^T & \widetilde{\boldsymbol{\Theta}}_{\Delta,t}^T & \widetilde{\boldsymbol{T}}_t^T & \widetilde{\boldsymbol{T}}_{\Delta,t}^T \end{bmatrix}^T \tag{5.95}$$

$$\begin{bmatrix} \breve{\boldsymbol{\Theta}}_t^T & \breve{\boldsymbol{\Theta}}_{\Delta,t}^T & \breve{\boldsymbol{T}}_t^T & \breve{\boldsymbol{T}}_{\Delta,t}^T \end{bmatrix}^T = \breve{\boldsymbol{x}}_t \tag{5.96}$$

$$\begin{bmatrix} \widetilde{\boldsymbol{\Theta}}_t^T & \widetilde{\boldsymbol{\Theta}}_{\Delta,t}^T & \widetilde{\boldsymbol{T}}_t^T & \widetilde{\boldsymbol{T}}_{\Delta,t}^T \end{bmatrix}^T = \widetilde{\boldsymbol{x}}_t \tag{5.97}$$

doi:10.6342/NTU202202468

The state $\breve{\boldsymbol{x}}_t$ is the filtered state after the direct LQR tracking. As shown in Equation (5.95), the tracked state target $\bar{\boldsymbol{x}}_t$ is composed of the position terms estimated from the OCR-UKF and the velocity terms filtered by the direct LQR tracking. Afterward, the accurate smoothed result can be regarded as the solution of Equation (5.98):

$$\min_{\boldsymbol{u}_t} \frac{1}{2}\left[\sum_{t=0}^{N-1}(\boldsymbol{x}_t-\bar{\boldsymbol{x}}_t)^T\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_t}\left(\boldsymbol{x}_t-\bar{\boldsymbol{x}}_{t_t}\right)+\boldsymbol{u}_t{}^T\boldsymbol{R}_{lqr2}\boldsymbol{u}_t\right] + \frac{1}{2}(\boldsymbol{x}_N-\bar{\boldsymbol{x}}_N)^T\boldsymbol{Q}_{lqr2,\breve{\boldsymbol{x}}_N}(\boldsymbol{x}_N-\bar{\boldsymbol{x}}_N) \tag{5.98}$$

$$\text{s.t. } \boldsymbol{x}_t=\boldsymbol{A}_{UKF}\boldsymbol{x}_{t-1}+\boldsymbol{B}_{UKF}\boldsymbol{u}_t,\ t=0,1,\dots,N-1 \text{ given } \boldsymbol{x}_0$$

To ensure the accuracy of the linearization, $\breve{\boldsymbol{x}}_t$ should be close to the real joint state which we can't access for estimation. To overcome it, the iterative LQR motion smoother is proposed as shown in **Algorithm 5.3**.

Algorithm 5.3: Iterative LQR motion smoothing

Input: a rough reference state $\hat{\boldsymbol{x}}_t$, forward kinematic function $h(\boldsymbol{x}_t)$, number of iterations N_{iter}

Output: the refined state $\bar{\bar{\boldsymbol{x}}}_t$

$\hat{\boldsymbol{z}}_t \leftarrow h(\hat{\boldsymbol{x}}_t)$

$\breve{\boldsymbol{x}}_t \leftarrow Joint\_Independent\_Tracking(\hat{\boldsymbol{x}}_t)$ – solve (5.87)

$\tilde{\boldsymbol{x}}_t \leftarrow \breve{\boldsymbol{x}}_t$

for $i \leftarrow 1$ to N_{iter} **do**

  $\boldsymbol{C}_{\tilde{\boldsymbol{x}}_t} \leftarrow \frac{\partial h}{\partial \boldsymbol{x}_t}|_{\boldsymbol{x}_t=\tilde{\boldsymbol{x}}_t}$

  $\tilde{\boldsymbol{x}}_t \leftarrow \tilde{\boldsymbol{x}}_t + pinv(\boldsymbol{C}_{\tilde{\boldsymbol{x}}_t})\cdot(\hat{\boldsymbol{z}}_t - h(\tilde{\boldsymbol{x}}_t))$

  $\bar{\boldsymbol{x}}_t \leftarrow \left[\tilde{\boldsymbol{\theta}}_t^T \quad \tilde{\boldsymbol{\theta}}_{\Delta,t}^T \quad \tilde{\boldsymbol{T}}_t^T \quad \tilde{\boldsymbol{T}}_{\Delta,t}^T\right]^T$

  $\bar{\bar{\boldsymbol{x}}}_t \leftarrow Joint\_Dependent\_keypoint\_Tracking(\bar{\boldsymbol{x}}_t)$ – solve (5.98)

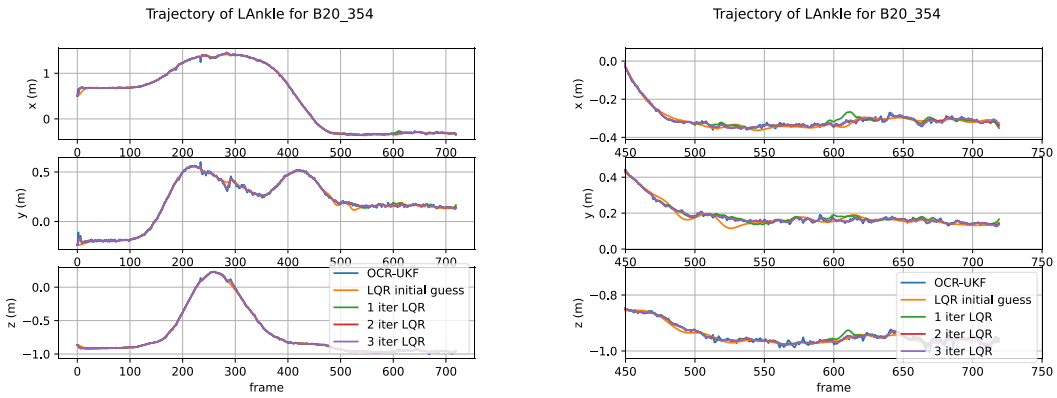
  $\tilde{\boldsymbol{x}}_t \leftarrow \bar{\bar{\boldsymbol{x}}}_t$

end for

return $\bar{\bar{\boldsymbol{x}}}_t$

doi:10.6342/NTU202202468

To guide $\check{\boldsymbol{x}}_t$ close to the real state, the joint-dependent state tracking with Equation (5.87) is set as $\check{\boldsymbol{x}}_t$ for the initial guess first. Since $\check{\boldsymbol{x}}_t$ is close to the real state with the initial guess, $\boldsymbol{C}_{\check{\boldsymbol{x}}_t}$ would also be close to the Jacobian at the real state. Therefore, the updated state $\bar{\bar{\boldsymbol{x}}}_t$ solved with Equation (5.98) can approach the real state. As $\check{\boldsymbol{x}}_t$ reach the real state, $\bar{\bar{\boldsymbol{x}}}_t$ would equal to $\check{\boldsymbol{x}}_t$ and not be changed anymore. To limit the operation time, the number of iterations is limited at N_{iter}. Usually, $N_{iter} = 3$ is enough, as shown in Figure 5.27. The estimation result after OCR-UKF and iterative LQR motion smoother would be called as PostSK.



(a) (b)

Figure 5.27: The result of iterative LQR motion smoother
(a) the overview of the trajectory of LAnkle for B20_354
(b) the zoomed in trajectory of LAnkle for B20_354

doi:10.6342/NTU202202468

Chapter 6
Simulation and Experimental Results and Analysis

In this chapter, the simulations and experiments are conducted to demonstrate and verify the applicability of the proposed system. In Section 6.1, the overview of the processes in simulations and experiments is reviewed. To evaluate the proposed method, error metrics to measure the performance of motion estimation are described in Section 6.2. The setups and results of the simulation to evaluate the skeleton motion modification method proposed in Chapter 5 are presented in Section 6.3 and Section 6.4 individually. To show the overall human motion estimation in the real-world, the experiment setups and results are shown in Section 6.5 and Section 6.6. Moreover, to compare with deep-learning-based methods popular in recent years, an evaluation tested on Human3.6M is demonstrated in Section 6.7.

doi:10.6342/NTU202202468

6.1 Overview of the Procedures of the Simulations and Experiments

To demonstrate the proposed method, both simulation and experiment will be shown in this chapter. In the simulation, the full processes to obtain the estimation motion from the rough and noisy skeleton mentioned in Chapter 5 will be evaluated as shown in Figure 6.1.

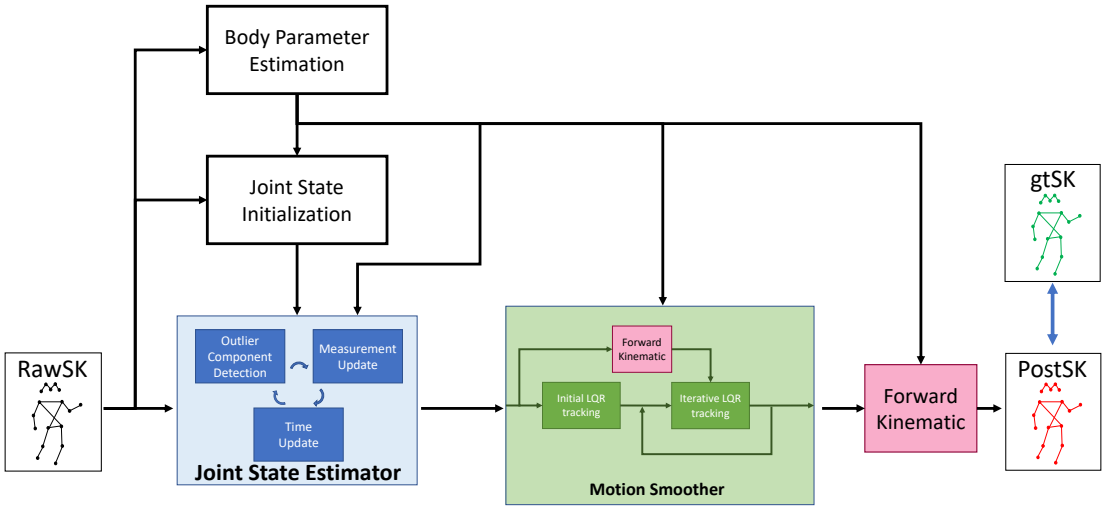
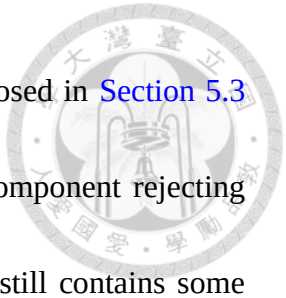


Figure 6.1: Flow chart for simulations

The rough skeleton (RawSK) will first pass through the body parameter estimation described in Section 5.2 to estimate the link parameters of a human skeleton. Next, the proposed method will use the estimated body parameters and the first few frames of the RawSK to reckon an approximate joint state for the initialization of the joint state estimator as Section 5.4 said.

doi:10.6342/NTU202202468



After the joint state initialization, the joint state estimator proposed in Section 5.3 will reduce the noise and even reject the outlier with the outlier-component rejecting mechanism. Hereafter, since the output of the joint state estimator still contains some detectable high-frequency noises, the proposed motion smoother with iterative LQR tracking mentioned in Section 5.5 will be applied to reduce the velocity and acceleration errors further. Finally, after the motion smoothing, the forward kinematic function will transform the estimation results from the joint space into the skeleton keypoint positions. The estimation results (PostSK) can be compared with the ground truth (gtSK) and evaluated its performance.

For the experiment, the whole procedures to obtain the estimated motion in real-world will be examined as shown in Figure 6.2. Not only the processes to modify the 3D raw skeletons but the processes to acquire the RawSKs from the multi-view system mentioned in Chapter 4 will also be investigated. The target human motion will be captured as videos with the proposed multi-view system presented in Section 4.1. The well-known 2D skeleton detection model, AlphaPose, will extract the 2D target skeletons in every view. With the 3D reconstruction method shown in Section 4.3, the RawSKs will be formed as the input of the 3D skeleton modification module which is the same processes as the simulation procedures.

doi:10.6342/NTU202202468

Besides the reconstructed results, in the experiment, the influence of the properties of the 3D reconstructed AlphaPose skeletons is also one of the points needed to be analyzed. The influence will directly determine the practical values of the proposed method.

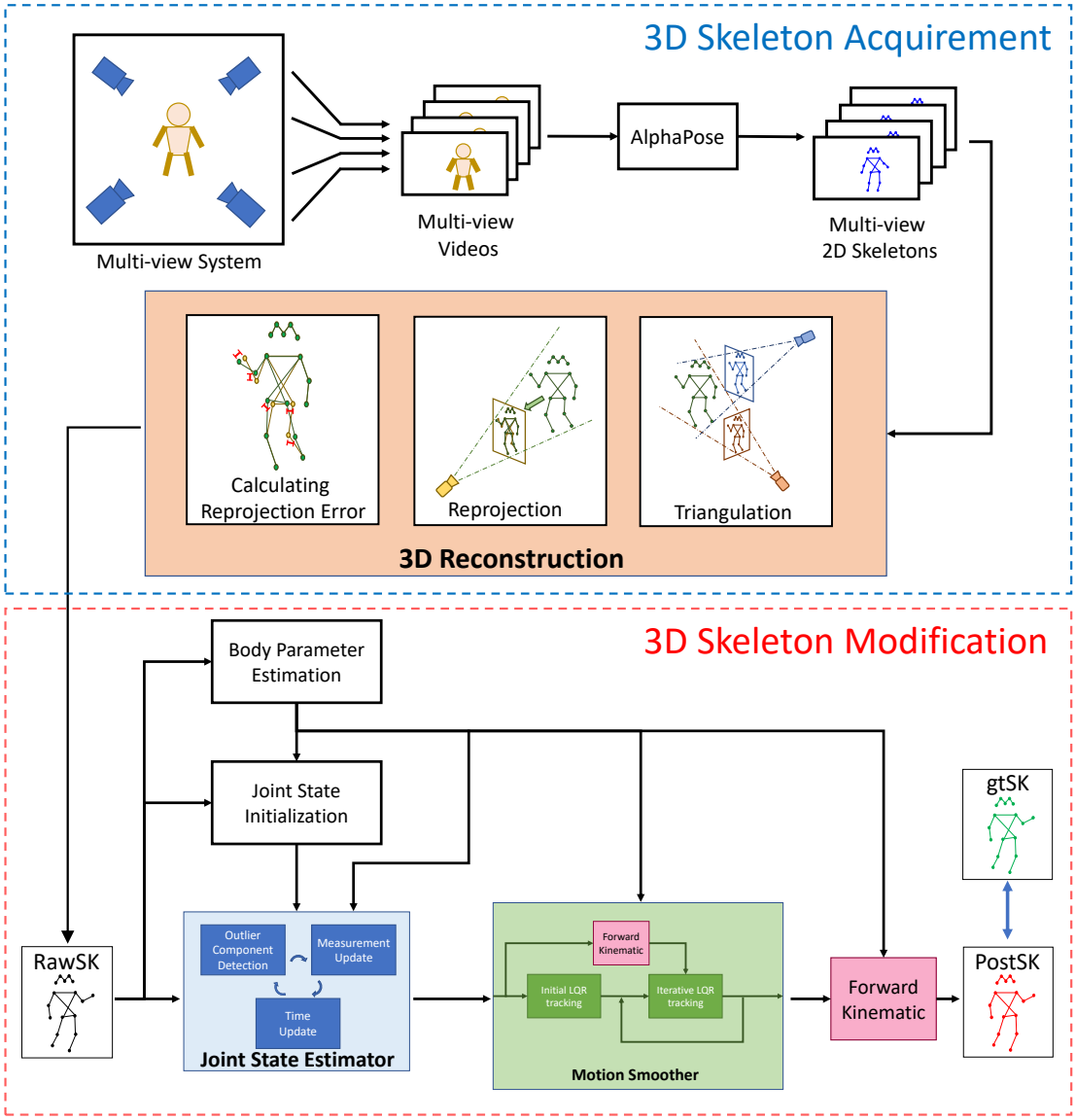


Figure 6.2: Flow chart for experiments

doi:10.6342/NTU202202468

6.2 Evaluation Metrics

Referring to [12: Wang et al. 2021], a common metric to quantify the 3D human pose estimation error is **MPJPE** (Mean Per Joint Position Error), which is calculated by:

$$E_{MPJPE}(\mathcal{S}) = \frac{1}{N}\sum_{t=1}^{N}\left[\frac{1}{N_s}\sum_{i=0}^{N_s-1}\left\|\boldsymbol{p}_{i,\mathcal{S}}(t) - \boldsymbol{p}_{i,gt}(t)\right\|_2\right] \tag{6.1}$$

where $\mathcal{S}$ denotes the corresponding skeleton to evaluate, $\boldsymbol{p}_{i,\mathcal{S}}(t) \in \mathbb{R}^3$ and $\boldsymbol{p}_{i,gt}(t) \in \mathbb{R}^3$ are the positions of i-th keypoint in $\mathcal{S}$ and ground truth skeleton at frame t, N is the frame number of the motion sequence, N_s is the keypoint number to compare.

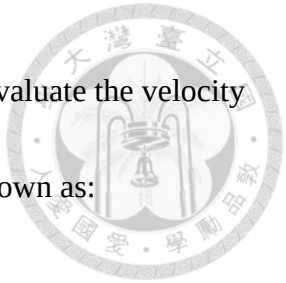
Apart from the position errors, the higher-order metrics for velocity and acceleration are used in this thesis as well since human motions are not only determined by its pose. To acquire the higher-order information, the discrete derivative is applied as Equation (6.9) and Equation (6.10):

$$\boldsymbol{v}_{i,\mathcal{S}}(t) = \boldsymbol{p}_{i,\mathcal{S}}(t) - \boldsymbol{p}_{i,\mathcal{S}}(t-1), \quad set\ \boldsymbol{p}_{i,\mathcal{S}}(0) = \boldsymbol{p}_{i,\mathcal{S}}(1) \tag{6.2}$$

$$\boldsymbol{v}_{i,gt}(t) = \boldsymbol{p}_{i,gt}(t) - \boldsymbol{p}_{i,gt}(t-1), \quad set\ \boldsymbol{p}_{i,gt}(0) = \boldsymbol{p}_{i,gt}(1) \tag{6.3}$$

where $\boldsymbol{v}_{i,\mathcal{S}}(t) \in \mathbb{R}^3$ and $\boldsymbol{v}_{i,gt}(t) \in \mathbb{R}^3$ are the velocities of i-th keypoint in $\mathcal{S}$ and ground truth skeleton at frame t.

doi:10.6342/NTU202202468



According to the quantification method of MPJPE, a metric to evaluate the velocity error of a skeleton $\mathcal{S}$, **MPJVE** (Mean Per Joint Velocity Error), is shown as:

$$E_{MPJVE}(\mathcal{S}) = \frac{1}{N}\sum_{t=1}^{N}\left[\frac{1}{N_s}\sum_{i=0}^{N_s-1}\left\|\boldsymbol{v}_{i,\mathcal{S}}(t) - \boldsymbol{v}_{i,gt}(t)\right\|_2\right] \tag{6.4}$$

Similar to MPJVE, the acceleration metric, **MPJAE** (Mean Per Joint Acceleration Error), can be calculated by:

$$E_{MPJAE}(\mathcal{S}) = \frac{1}{N}\sum_{t=1}^{N}\left[\frac{1}{N_s}\sum_{i=0}^{N_s-1}\left\|\boldsymbol{a}_{i,\mathcal{S}}(t) - \boldsymbol{a}_{i,gt}(t)\right\|_2\right] \tag{6.5}$$

where $\boldsymbol{a}_{i,\mathcal{S}}(t) \in \mathbb{R}^3$ and $\boldsymbol{a}_{i,gt}(t) \in \mathbb{R}^3$ are the accelerations of i-th keypoint in $\mathcal{S}$ and ground truth skeleton at frame t which can be calculated as:

$$\boldsymbol{a}_{i,\mathcal{S}}(t) = \boldsymbol{v}_{i,\mathcal{S}}(t) - \boldsymbol{v}_{i,\mathcal{S}}(t-1)\ ,\quad set\ \boldsymbol{v}_{i,\mathcal{S}}(0) = \boldsymbol{v}_{i,\mathcal{S}}(1) \tag{6.6}$$

$$\boldsymbol{a}_{i,gt}(t) = \boldsymbol{v}_{i,gt}(t) - \boldsymbol{v}_{i,\mathcal{S}}(t-1)\ ,\quad set\ \boldsymbol{v}_{i,gt}(0) = \boldsymbol{v}_{i,gt}(1) \tag{6.7}$$

These three metrics are the main metrics to evaluate the performance of the proposed method from the views of positions (MPJPE), velocity (MPJVE) and acceleration (MPJAE).

doi:10.6342/NTU202202468

6.3 Simulation Setups

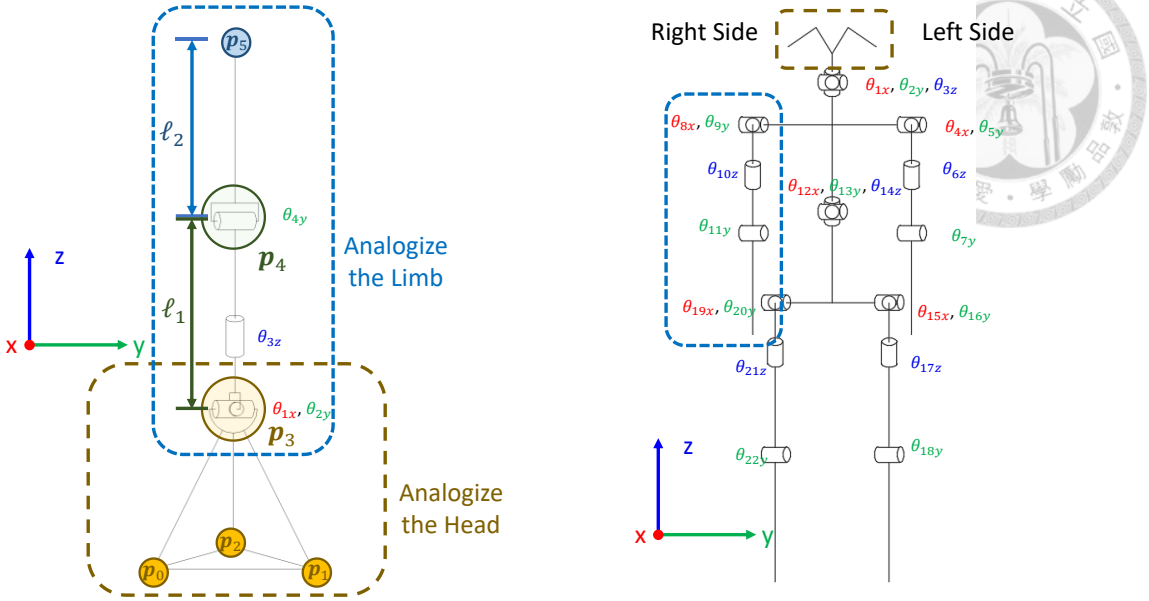
In the simulation, the main purpose is to investigate the properties of the proposed skeleton modifying method from different views. The ground truth motions will be generated with hand-craft ideal joint state and body parameters. Then, the ground truth keypoint positions will be obtained by passing through the forward kinematic function. To produce the RawSKs as the inputs, additional noises will be added to the ground truth keypoint position. These noises are used to simulate the real noise caused by occlusion, frame dropping and imperfect detection from the keypoint detector. Therefore, the noises will contain basic **noises**, **outliers** and some **missing data** to reproduce the real situation.

6.3.1 Kinematic model Setups

In the simulation, to reduce the factors influencing the estimation result, a simplified skeleton (SimSK) is proposed to analogize the human skeleton, as shown in Figure 6.3. There are 6 keypoints and 4 joints in a SimSK. The numbers of estimated body parameters are 4 for the head and 2 for the limb shown in Table 6.1.

The simplified skeleton would be applied to replace the human skeleton in this simulation to reduce the considered variables.

doi:10.6342/NTU202202468



(a) (b)

Figure 6.3: Analogy of SimSK and human skeleton
(a) The kinematic model of SimSK
(b) The kinematic model of human skeleton

In Figure 6.3 (a), the head part of SimSK labeled in yellow is a rigid-body structure and is used to analogize the head part of the human skeleton in Figure 6.3 (b). There are 4 points in the head part, which is the minimum number of the points to apply the 3D transformation estimation described in Section 3.6. The transformation state between the head part and the world coordinate will also be estimated as the head-world transformation in the human skeleton.

For the limb part labeled in blue in Figure 6.3 (a), it's used to analogize the limb structure in the human skeleton. From $\boldsymbol{p}_3$ to $\boldsymbol{p}_5$, the three keypoints are used to represent the root point (shoulder or hip), middle point (elbow or knee) and endpoint (wrist or ankle) in a limb of a human skeleton. They have corresponding kinematic structure for the joint state.

doi:10.6342/NTU202202468

Consequentially, there are six body parameters to describe the SimSK. As Table 6.1 shown, ℓ_1 and ℓ_2 are the parameters represent the lengths of both sections in the limb part. For the head part, $\left({}^{head}\boldsymbol{p}_0, {}^{head}\boldsymbol{p}_1, {}^{head}\boldsymbol{p}_2, {}^{head}\boldsymbol{p}_4\right)$ are the parameters to represent the keypoint positions in the rigid-body coordinate $\{head\}$.

Table 6.1

The definition of the body parameters in a SimSK

Symbol	Definition
ℓ_1	Length of Upper section of Limb ($\in \mathbb{R}$)
ℓ_2	Length of Lower section of Limb ($\in \mathbb{R}$)
${}^{head}\boldsymbol{p}_0$	Position of $\boldsymbol{p}_0$ in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_1$	Position of $\boldsymbol{p}_1$ in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_2$	Position of $\boldsymbol{p}_2$ in $\{head\}$ ($\in \mathbb{R}^3$)
${}^{head}\boldsymbol{p}_3$	Position of $\boldsymbol{p}_3$ in $\{head\}$ ($\in \mathbb{R}^3$)

In the simulation, the ideal body parameters are set as:

$$\begin{cases} \ell_1 = 0.4\ (m) \\ \ell_2 = 0.3\ (m) \end{cases} \tag{6.8}$$

$$\begin{cases} {}^{head}\boldsymbol{p}_0 = [0 \quad 0 \quad 0]^T\ (m) \\ {}^{head}\boldsymbol{p}_1 = [0.2 \quad 0 \quad 0]^T\ (m) \\ {}^{head}\boldsymbol{p}_2 = [0 \quad 0.2 \quad 0]^T\ (m) \\ {}^{head}\boldsymbol{p}_3 = [0 \quad 0 \quad 0.2]^T\ (m) \end{cases} \tag{6.9}$$

doi:10.6342/NTU202202468

When the state is equal to a zero vector, the keypoints of ideal SimSK is visualized as Figure 6.4 shown.

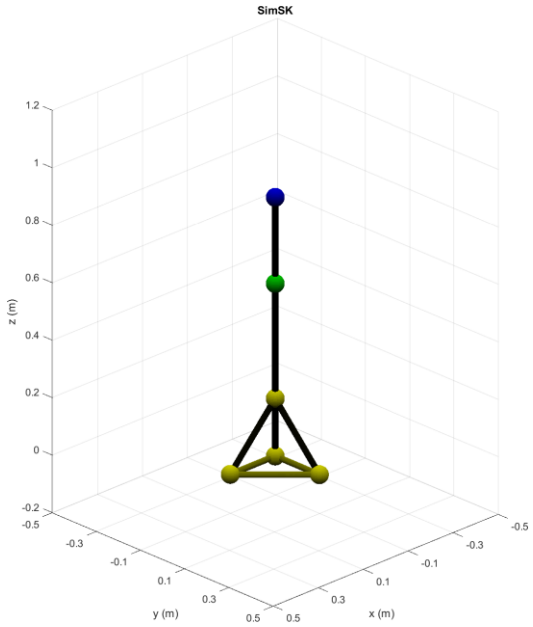


Figure 6.4: The SimSK with given ideal body parameters

6.3.2 Tested Motions

In the simulation, the ideal state $\boldsymbol{x}_{ideal,\,t}$ for the SimSK is defined with following variables:

$$\begin{cases} p_{0,x} = \cos \omega t \\ p_{0,y} = sin\, \omega t \\ p_{0,z} = sin \dfrac{1}{2} \omega t \end{cases}, \quad \begin{cases} \alpha = 0 \\ \beta = 0 \\ \gamma = 0 \end{cases} \tag{6.10}$$

$$\begin{cases} \theta_{1,x} = {}^{\pi}/_{2} \cdot sin\, \omega t \\ \theta_{2,y} = {}^{\pi}/_{2} \cdot sin\, \omega t \\ \theta_{3,z} = {}^{\pi}/_{2} \cdot sin\, 2\omega t \\ \theta_{4,y} = {}^{\pi}/_{4} \cdot sin \dfrac{3}{2} \omega t + {}^{\pi}/_{2} \end{cases} \tag{6.11}$$

doi:10.6342/NTU202202468

The trajectories of the state values are illustrated in Figure 6.5:

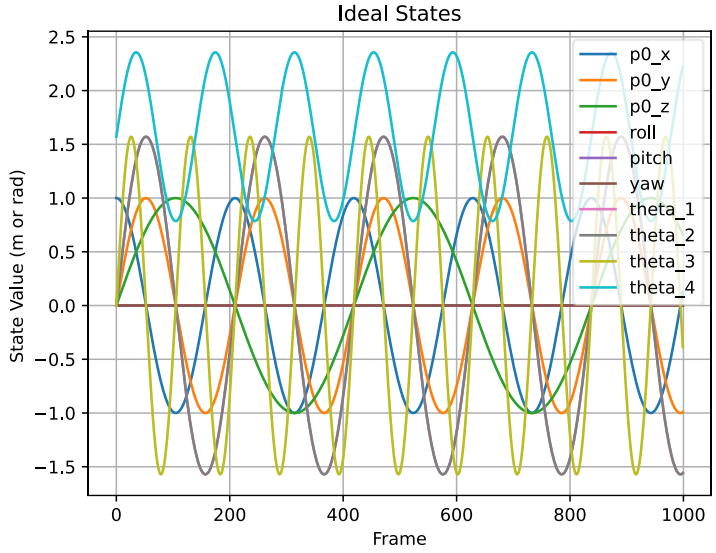


Figure 6.5: The ideal state of SimSK in simulation

$(p_{0,x}, p_{0,y}, p_{0,z})$ is the position of $\boldsymbol{p}_0$. (α, β, γ) are the roll-pitch-yaw angle of the head. $\theta_{1,x}$ to $\theta_{4,y}$ are the joint angle in the SimSK. ω is the motion speed factor and set as 0.03.

Therefore, the ideal state $\boldsymbol{x}_{ideal,\,t}$ is:

$$\boldsymbol{x}_{ideal,\,t} = \left[\boldsymbol{T}_t^T \quad \boldsymbol{\Theta}_t^T \quad \boldsymbol{T}_{\Delta,t}^T \quad \boldsymbol{\Theta}_{\Delta,t}^T\right]^T \in \mathbb{R}^n, \qquad n = 20 \tag{6.12}$$

$$\begin{cases} \boldsymbol{T}_t = [p_{0,x} \quad p_{0,y} \quad p_{0,z} \quad \alpha \quad \beta \quad \gamma]^T \\ \boldsymbol{\Theta}_t = [\theta_{1,x} \quad \theta_{2,y} \quad \theta_{3,z} \quad \theta_{4,y}]^T \\ \boldsymbol{T}_{\Delta,t} = \boldsymbol{T}_t - \boldsymbol{T}_{t-1} \\ \boldsymbol{\Theta}_{\Delta,t} = \boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t-1} \end{cases} \tag{6.13}$$

With the defined ideal joint state $\boldsymbol{x}_{ideal,\,t}$ in Equation (6.10)-(6.11), the ideal motion can be generated with forward kinematic function $h(\cdot)$ as:

$$\boldsymbol{z}_{ideal,\,t} = [\boldsymbol{p}_0^T \quad \boldsymbol{p}_1^T \quad \boldsymbol{p}_2^T \quad \boldsymbol{p}_3^T \quad \boldsymbol{p}_4^T \quad \boldsymbol{p}_5^T]^T = h(\boldsymbol{x}_{ideal,\,t}) \tag{6.14}$$

Therefore, the ideal motion of the SimSK can be illustrated in Figure 6.6.

doi:10.6342/NTU202202468

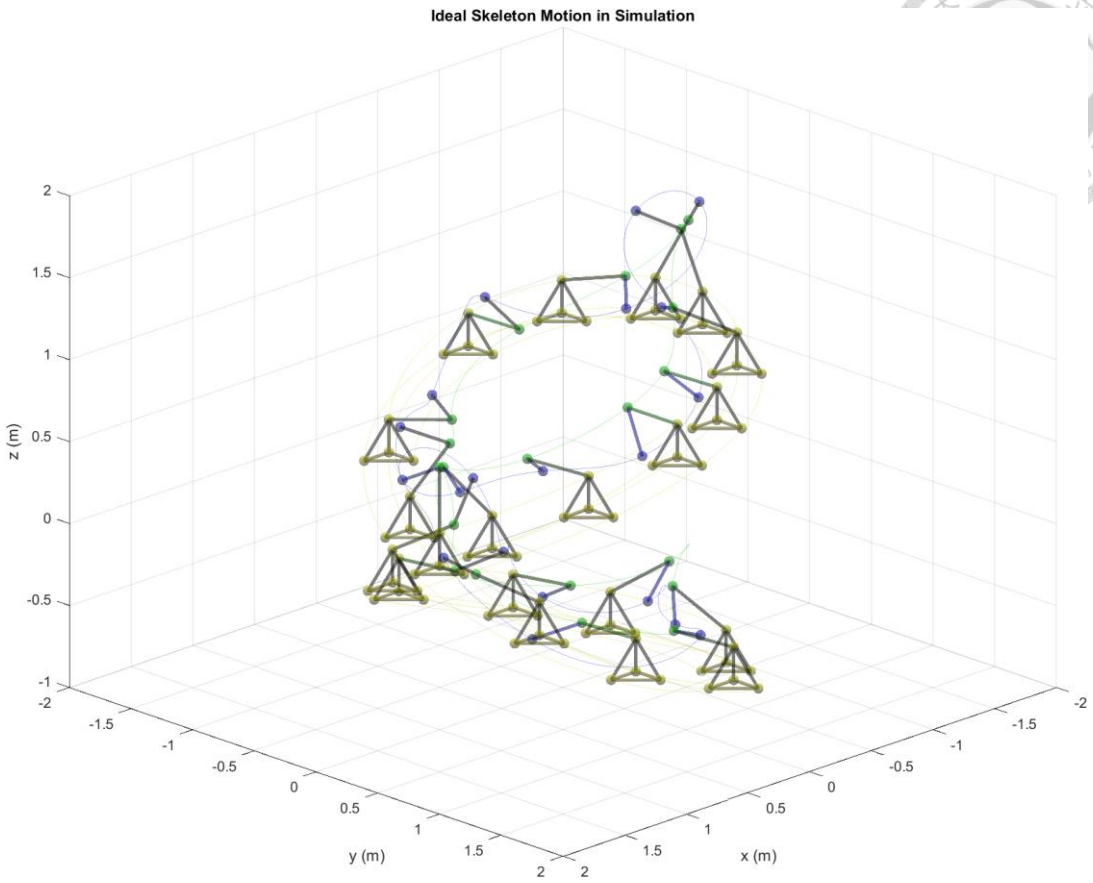


Figure 6.6: The ideal motion (gtSK) of SimSK in simulation

The ideal motion will be regarded as the ground truth (gtSK) in the simulation, i.e:

$$\boldsymbol{z}_{gt,t} = \boldsymbol{z}_{ideal,t} \in \mathbb{R}^{18} \tag{6.15}$$

To demonstrate the proposed estimation method, the RawSK will be generated with the ground truth. Since the RawSK is used to simulate the raw noisy motion sampled from the real world, it should contain the three defects like the real raw data: **basic noises**, **outliers** and **missing data**.

doi:10.6342/NTU202202468

For the basic noises, we use a white noise $\boldsymbol{w}_t$ multiplied with a gain G_{bn} to simulate the small observation noise as Equation (6.16).

$$\boldsymbol{z}_{bn,t} = \boldsymbol{z}_{gt,t} + G_{bn,t} \cdot \boldsymbol{w}_t \in \mathbb{R}^{18} \tag{6.16}$$

$$\boldsymbol{w}_t \sim \mathcal{N}(0,1) \tag{6.17}$$

$$G_{bn,t} = 0.03 \tag{6.18}$$

Then, for the outliers caused by the imperfect detection of AlphaPose and the 3D reconstruction, we first design a probability $P_{outlier\_start}$ for every keypoint at each frame as the start of the outlier intervals.

$$P_{outlier\_start} = 0.025 \tag{6.19}$$

Once an outlier interval is chosen to start in the i-th keypoint at the frame t, a length will be sampled as the duration of the outlier interval from Poisson distribution:

$$N_{out,i,t} \sim Pois(\lambda_{out}) \tag{6.20}$$

$$\lambda_{out} = 2 \tag{6.21}$$

In the outlier interval, the raw motion will be added to the same random vector in the whole interval where the random vector is calculated by:

$$\boldsymbol{v}_{i,t} = G_{out,i,t} \cdot \boldsymbol{n}_{i,t} \in \mathbb{R}^3 \tag{6.22}$$

$$\boldsymbol{n}_{i,t} = \frac{\boldsymbol{u}_{i,t}}{\|\boldsymbol{u}_{i,t}\|} \tag{6.23}$$

where $G_{out,i,t}$ is sampled from a uniform distribution in $[0.3, 0.8]$, and $\boldsymbol{u}_{i,t} \in \mathbb{R}^3$ is a random vector whose elements are all sampled from a uniform distribution in $[-1, 1]$.

doi:10.6342/NTU202202468

For the outlier part, since the detection results of AlphaPose are highly pose-dependent, the outliers usually sustain in a few frames because the human pose is highly similar in the interval. Likewise, the outlier values in the outlier interval are also similar. Thus, the outlier situations can be simulated with the approach above.

With this approach, the overall outlier probability $P_{outlier}$ can be calculated as Equation since $E[Pois(\lambda_{out})] = \lambda_{out}$.

$$P_{outlier} = P_{outlier\_start} \cdot \lambda_{out} = 0.05 \tag{6.24}$$

Then, for the missing data, similar to outliers, the missing data usually sustain a few frames but longer. Therefore, we can use the same approach. First, set the probability P_{miss_start} of the start of the missing interval as:

$$P_{miss\_start} = 0.005 \tag{6.25}$$

The duration of the missing interval in the i-th keypoint at the frame t is also sampled from Poisson distribution:

$$N_{miss,i,t} \sim Pois(\lambda_{miss}) \tag{6.26}$$

$$\lambda_{miss} = 10 \tag{6.27}$$

All the keypoint values in the missing interval will be removed to simulate the missing data situation in the real data. Finally, the overall missing probability is calculated as:

$$P_{miss} = P_{miss\_start} \cdot \lambda_{miss} = 0.05 \tag{6.28}$$

doi:10.6342/NTU202202468

The stages of generating RawSKs are illustrated in Figure 6.7. After obtaining the ideal motion, the basic noises are added. The outlier intervals with their additional outlier vectors $\boldsymbol{v}_{i,t}$ are joined. Last, to simulate the missing data problem, the missing interval are sampled and wiped out the values.

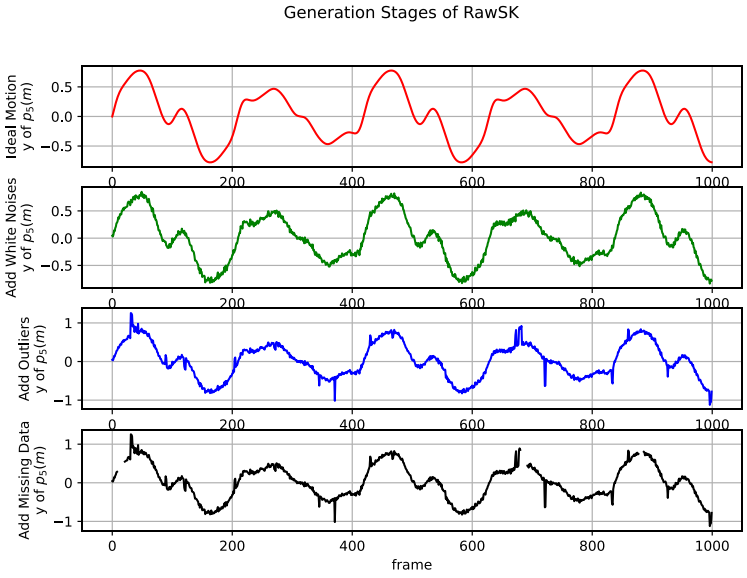


Figure 6.7: Generation stages of RawSK

doi:10.6342/NTU202202468

Eventually, the RawSK motion in the simulation is illustrated in Figure 6.8.

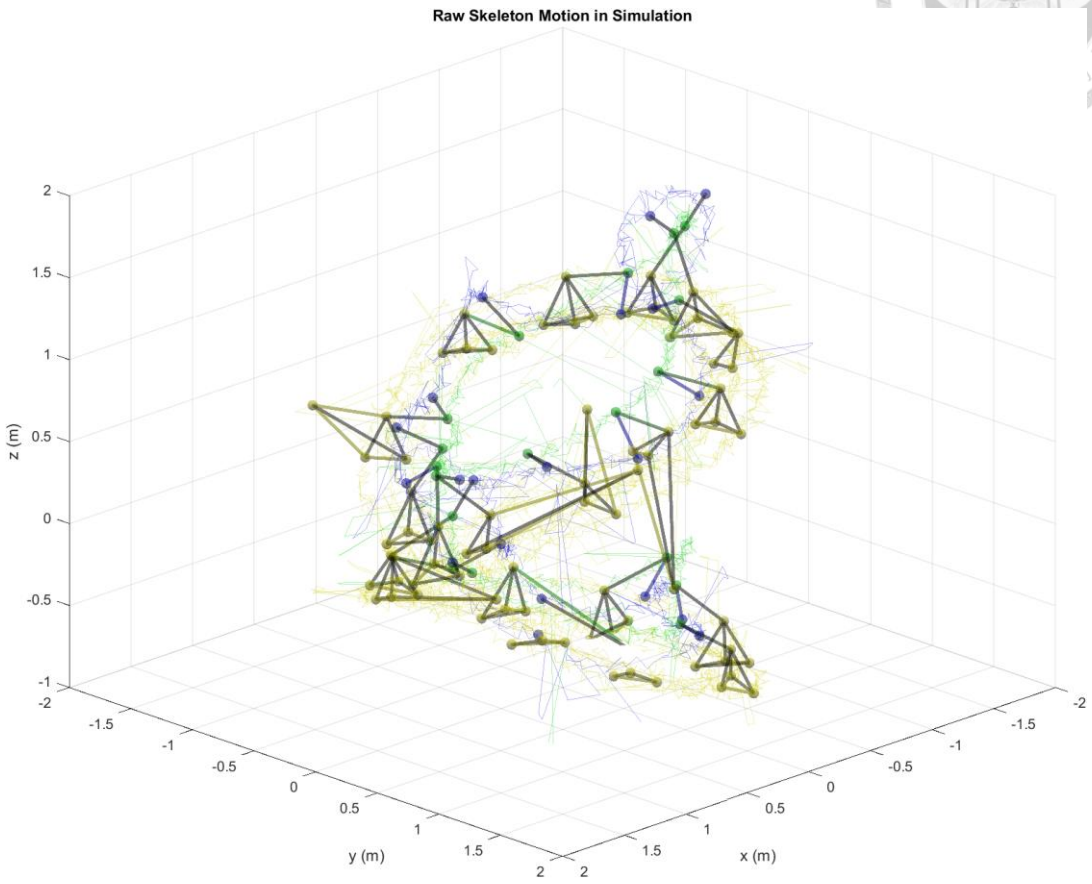


Figure 6.8: The raw motion of SimSK in simulation

doi:10.6342/NTU202202468

6.3.3 Parameter Settings

In the simulation, the parameters set for the proposed estimation method are shown as Table 6.2 and Table 6.3:

Table 6.2
The Parameter Settings for the Joint State Estimator in the Proposed Method in the Simulation

Joint State Estimator:			
	Symbol	Value	Meaning
	$\boldsymbol{P}_0$	$10^{-1} \cdot \boldsymbol{I}_{20}$	Initial state covariance
	$\boldsymbol{Q}_{UKF}$	$\begin{bmatrix} \boldsymbol{0}_{10\times10} & \boldsymbol{0}_{10\times10} \\ \boldsymbol{0}_{10\times10} & 10^{-5} \cdot \boldsymbol{I}_{10} \end{bmatrix}$	Process noise covariance
	$\boldsymbol{R}_{UKF}$	$10^{-5} \cdot \boldsymbol{I}_{18}$	Measurement noise covariance
	k_{out}	0.008	Outlier rejecting coefficient
	$\boldsymbol{Q}_{in,N_{out}}$	$10^{10} \cdot \boldsymbol{I}_{20}$	State weights for inlier interpolation
	$\boldsymbol{R}_{in}$	$\boldsymbol{I}_{10}$	Input weights for inlier interpolation
	$\boldsymbol{x}_{lower}$	$-\infty \cdot \boldsymbol{1}_{20}$	State lower bound with KKT condition
	$\boldsymbol{x}_{upper}$	$\infty \cdot \boldsymbol{1}_{20}$	State upper bound with KKT condition
	c_{damp}	-0.1	Damper coefficient for body force

doi:10.6342/NTU202202468

Table 6.3

The Parameter Settings for the Motion Smoother in the Proposed Method in the Simulation

Motion Smoother:			
	Symbol	Value	Meaning
	$\boldsymbol{Q}_{lqr1}$	$\boldsymbol{I}_{20}$	State weights for initial LQR tracking
	$\boldsymbol{R}_{lqr1}$	$10^3 \cdot \boldsymbol{I}_{10}$	Input weights for initial LQR tracking
	$\boldsymbol{Q}_{lqr2}$	$\begin{bmatrix} \boldsymbol{I}_{12} & \boldsymbol{0}_{12\times6} \\ \boldsymbol{0}_{6\times12} & 50 \cdot \boldsymbol{I}_{6} \end{bmatrix}$	State weights for iterative LQR tracking
	$\boldsymbol{Q}_{lqr2,vel}$	$\begin{bmatrix} \boldsymbol{0}_{10\times10} & \boldsymbol{0}_{10\times10} \\ \boldsymbol{0}_{10\times10} & 10^4 \cdot \boldsymbol{I}_{10} \end{bmatrix}$	Velocity state weights for iterative LQR tracking
	$\boldsymbol{R}_{lqr2}$	$\begin{bmatrix} 10^3 \cdot \boldsymbol{I}_{6} & \boldsymbol{0}_{6\times4} \\ \boldsymbol{0}_{4\times6} & 10^2 \cdot \boldsymbol{I}_{4} \end{bmatrix}$	Input weights for iterative LQR tracking
	N_{iter}	3	Number of iterations for iterative LQR tracking

doi:10.6342/NTU202202468

6.4 Simulation Results and Analysis

6.4.1 Performance Analysis of Motion Estimation

As the proposed method shown in Figure 6.1, the first step of the skeleton modification is to estimate the body parameters of the skeleton (SimSK). The estimated results and the ground truth values of the body parameters are listed in Table 6.4. The maximum estimation error of the body parameters is in ${}^{head}\boldsymbol{p}_1$ and high to 46 mm. The overall body parameter estimation errors are from 0 to 46 mm. These errors are acceptable, however, may influence the motion estimation.

Table 6.4

The Estimated Body Parameters in the Simulation

Body Parameter	Ground Truth (m)	Estimated Value (m)
ℓ_1	0.4	0.409
ℓ_2	0.3	0.317
${}^{head}\boldsymbol{p}_0$	$[0 \quad 0 \quad 0]^T$	$[0 \quad 0 \quad 0]^T$
${}^{head}\boldsymbol{p}_1$	$[0.2 \quad 0 \quad 0]^T$	$[0.246 \quad 0 \quad 0]^T$
${}^{head}\boldsymbol{p}_2$	$[0 \quad 0.2 \quad 0]^T$	$[0.041 \quad 0.212 \quad 0]^T$
${}^{head}\boldsymbol{p}_3$	$[0 \quad 0 \quad 0.2]^T$	$[0.038 \quad 0.011 \quad 0.177]^T$

The estimated trajectories and errors of all keypoints in the SimSK are illustrated in Figure 6.9 and Figure 6.10.

doi:10.6342/NTU202202468

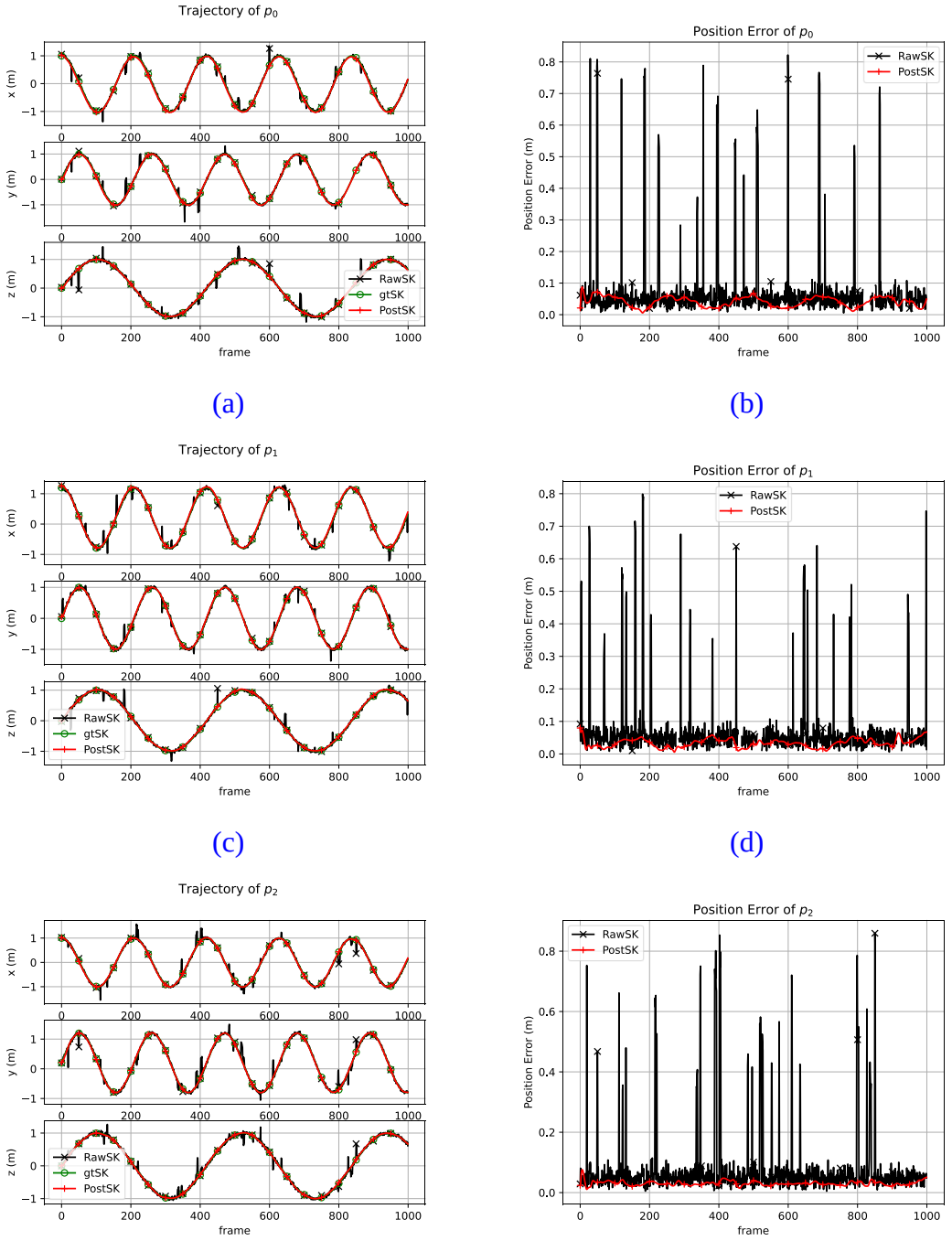


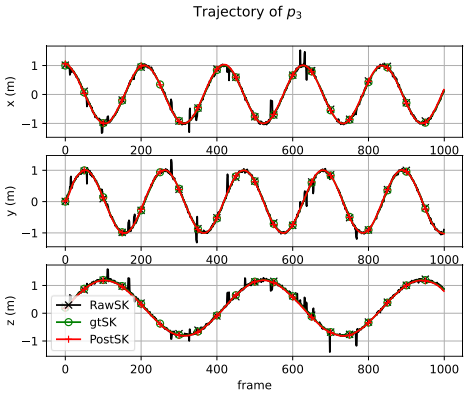
Figure 6.9: The estimated trajectory for $\boldsymbol{p}_0$, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ in Simulation

(a) Trajectory of $\boldsymbol{p}_0$ (b) Position Error of $\boldsymbol{p}_0$

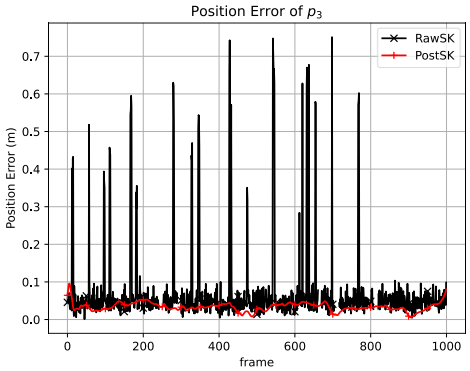
(c) Trajectory of $\boldsymbol{p}_1$ (d) Position Error of $\boldsymbol{p}_1$

(e) Trajectory of $\boldsymbol{p}_2$ (f) Position Error of $\boldsymbol{p}_2$

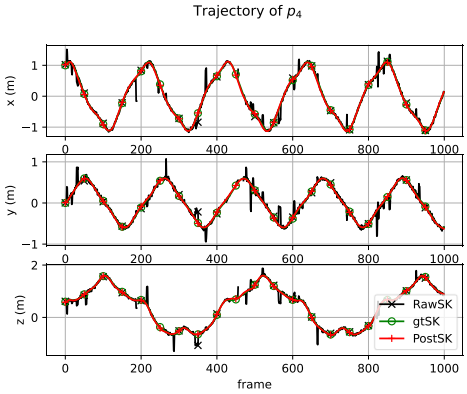
doi:10.6342/NTU202202468



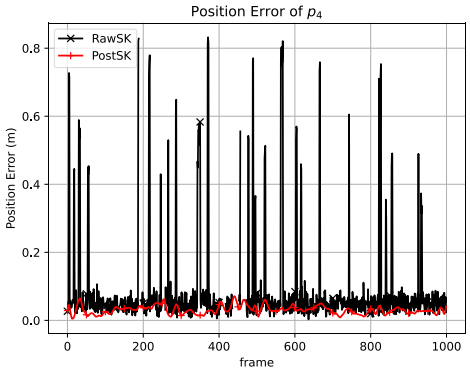
(a)



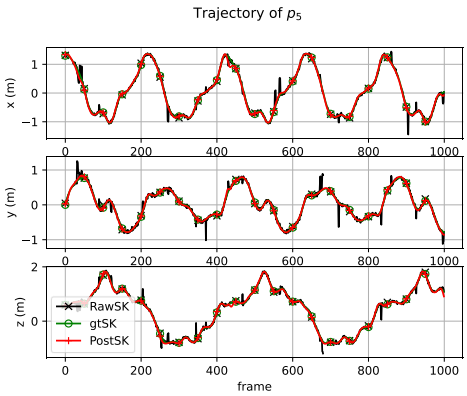
(b)



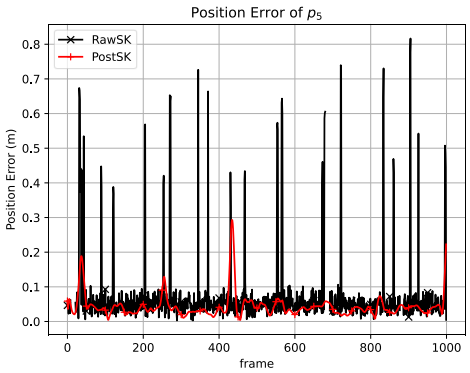
(c)



(d)



(e)



(f)

Figure 6.10: The estimated trajectory for $\boldsymbol{p}_3$, $\boldsymbol{p}_4$ and $\boldsymbol{p}_5$ in Simulation
(a) Trajectory of $\boldsymbol{p}_3$ (b) Position Error of $\boldsymbol{p}_3$
(c) Trajectory of $\boldsymbol{p}_4$ (d) Position Error of $\boldsymbol{p}_4$
(e) Trajectory of $\boldsymbol{p}_5$ (f) Position Error of $\boldsymbol{p}_5$

doi:10.6342/NTU202202468

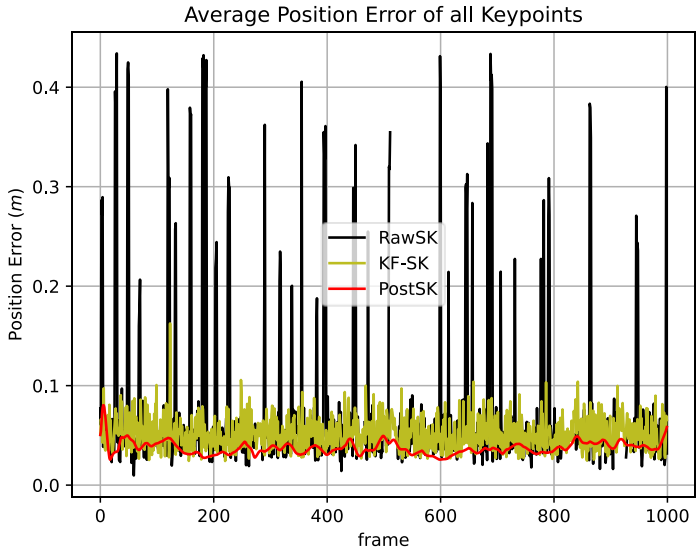
In Figure 6.9 and Figure 6.10, we can notice that the errors for most keypoints in PostSK are much lower than those in RawSK. It's because the outlier-component-rejecting mechanism we design in Section 5.3 works properly. In most frames, the errors are less relative to the outliers from the RawSK, which increases the robustness of the estimation results.

However, for the keypoint $\boldsymbol{p}_5$, there are huge errors occurring in few frames with the outliers. As $\boldsymbol{p}_5$ is the end point of the SimSK, there is lower correlations between $\boldsymbol{p}_5$ and other keypoints. In other words, there is more freedom and higher uncertainty for $\boldsymbol{p}_5$ given the information on the other keypoints. Therefore, it's harder to determine the outliers when the outliers occur in $\boldsymbol{p}_5$. It's the reason caused by the kinematic structure.

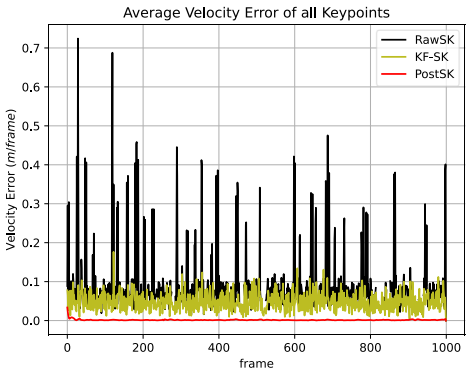
For the overall estimation errors, the average errors of the six keypoints are illustrated in Figure 6.11. There are three stages of the estimation in the proposed method: 1) RawSK, 2) the estimated skeleton after the joint state estimator (KF-SK), and 3) PostSK. As mentioned in Chapter 5, the task of the joint state estimator is to reject the outliers, fill the missing data and propose the joint state with better position performance. Then, the task of motion smoother is to reduce the high-order errors and further refine the skeleton position by considering the trade-off between state tracking and joint acceleration.

doi:10.6342/NTU202202468

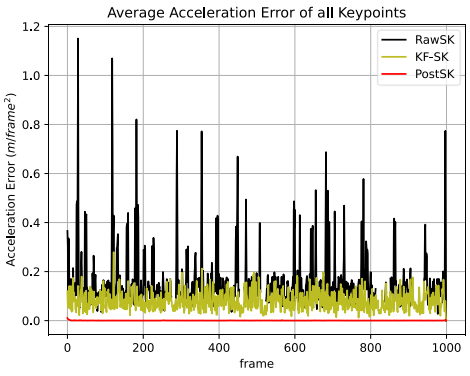
In Figure 6.11 (a), it's easy to observe that the KF-SK rejects the missing data and most of the outliers and then performs much better position errors. For the PostSK, the motion smoother lowers the position errors again in Figure 6.11 (a) and enhances the velocity and acceleration performance in Figure 6.11 (b) and (c).



(a)



(b)



(c)

Figure 6.11: The average error of all keypoints in simulation
(a) average position error (b) average velocity error (c) average acceleration error

doi:10.6342/NTU202202468

The precise improvement of MPJPE, MPJVE and MPJAE in different stages are listed in Table 6.5.

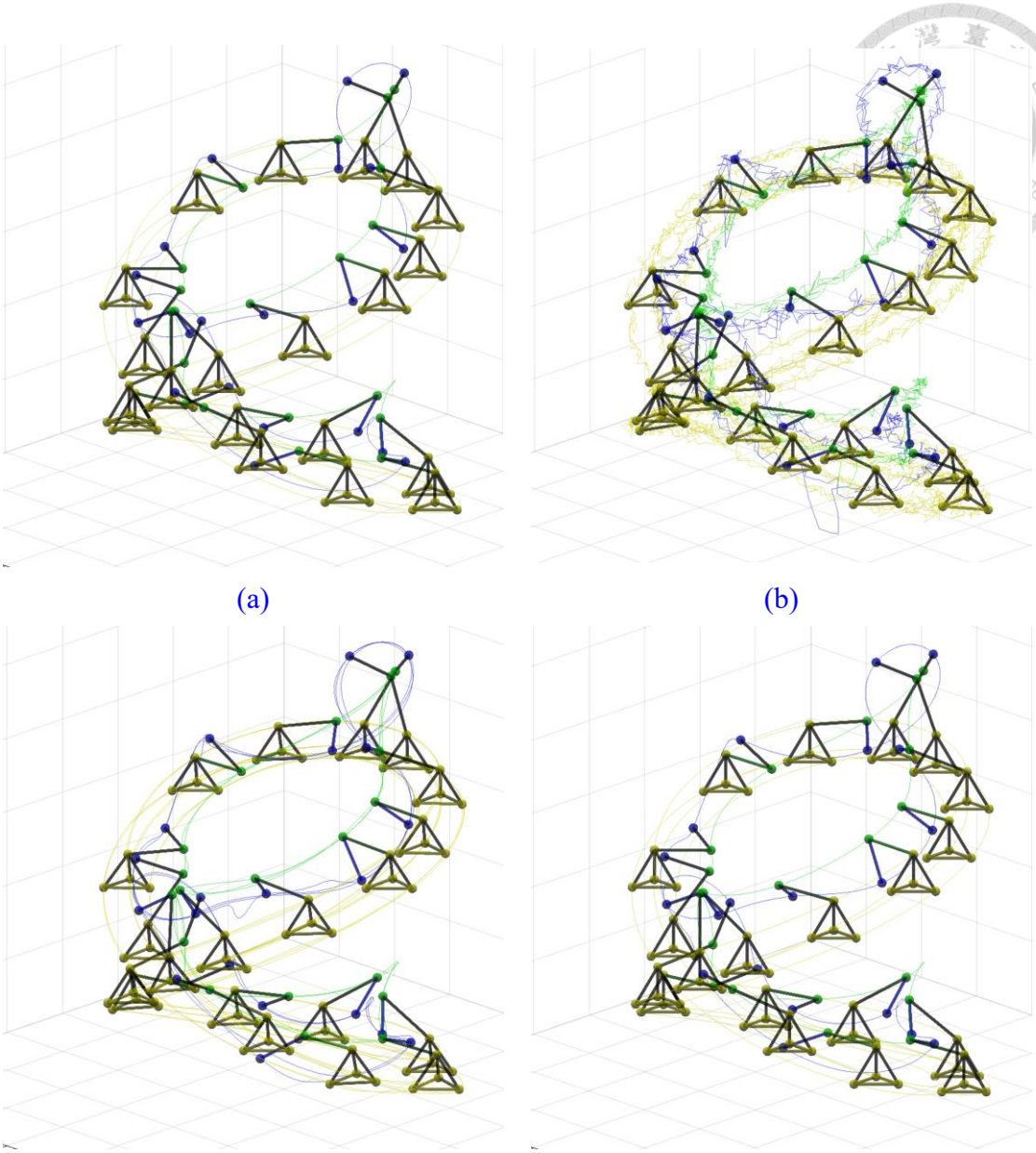
Table 6.5
Estimation Performance in the Simulation

Skeleton $\mathcal{S}$	$E_{MPJPE}(\mathcal{S})$ Unit: (mm)	$E_{MPJVE}(\mathcal{S})$ Unit: (mm/frame)	$E_{MPJAE}(\mathcal{S})$ Unit:(mm/frame^2)
RawSK	75.93	90.47	159.91
KF-SK	52.27	52.36	87.44
PostSK	35.37	2.27	0.56

The MPJPEs in each stage keep dropping in the whole estimation flow. The MPJVEs and MPJAEs decrease dramatically in proportion after the motion smoothing stage.

The estimated trajectories in each stage are displayed in Figure 6.12. In RawSK, there are lots of outliers and missing data. The overall trajectories are chaotic. In KF-SK, all the missing intervals are filled. Most of the outliers are rejected. But the trajectories are still rough and shaky. After the motion smoother, the trajectories of the PostSK are much smoother and look like the ground truth skeleton in Figure 6.12 (d).

doi:10.6342/NTU202202468



(a) (b)

(c) (d)

Figure 6.12: The estimated trajectories of every keypoint in different estimation stages (a) Raw skeleton (b) KF skeleton after the joint estimator (c) Post skeleton (d) Ground truth skeleton

doi:10.6342/NTU202202468

6.4.2 Effectiveness of OCR Joint State Estimator

In this section, the effectiveness of the outlier-component-rejecting joint state estimator will be discussed. The discussed topic is composed of two parts: 1) the effectiveness of OCR (outlier component rejection) and 2) the performance with different joint state estimators. There are four independent variables to investigate the performance under different RawSK properties: 1) probability of outliers $P_{outlier}$, 2) expected outlier interval duration λ_{out}, 3) probability of missing data P_{miss}, and 4) expected missing interval duration λ_{miss}. Since the RawSKs contain random variables, they will be run 10 times and be taken the average errors for each parameter setting. For the other parameters not set as the independent variable, they will remain in the same settings in Section 6.3.2.

1) Different probability of outliers $P_{outlier}$:

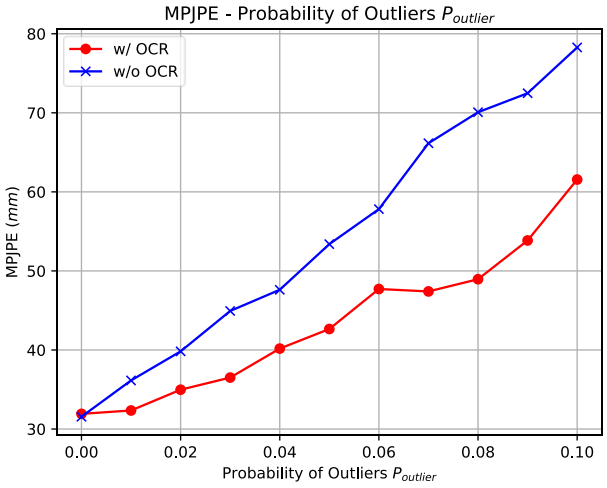
To test the effectiveness of outlier component rejection (OCR), there are two joint state estimators to be compared. The first one is the proposed method. The other one is the same as the proposed one but its outlier component rejecting mechanism is turned off, which means that the outlier rejecting coefficient k_{out} is set as 0.

As shown in Figure 6.13, the performance errors increase as the probability of outliers $P_{outlier}$ grows no matter with or without OCR. For the estimator without OCR, there are more false detections being regarded as the normal observation so that the

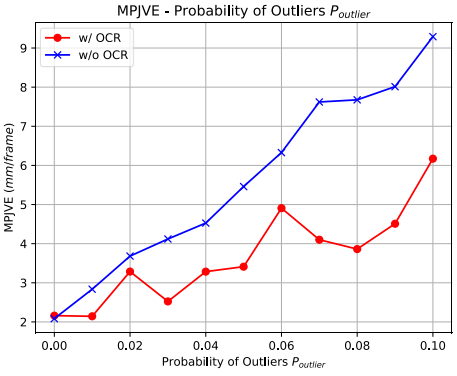
doi:10.6342/NTU202202468

estimation errors will raise. For the estimator with OCR, since the observation covariance will go up when the outliers are rejected, there is higher opportunities to regard the outliers as inliers when $P_{outlier}$ grows.

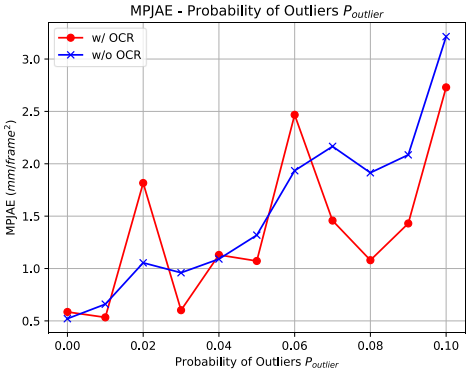
However, as Figure 6.13 (a) and (b) shown, the improvements by the OCR mechanism in MPJPE and MPJVE also increase when $P_{outlier}$ grows, which means that outlier resistance ability of the estimator with OCR is greater than the one without OCR.



(a)



(b)



(c)

Figure 6.13: The performance with or without OCR (outlier component rejection) in different probability of outliers $P_{outlier}$
(a) Position Error (b) Velocity Error (c) Acceleration Error

doi:10.6342/NTU202202468

For MPJAE in Figure 6.13 (c), there is no obvious trend for the improvements by the OCR mechanism. One of the inferences is that acceleration is pretty close to the ground truth after the motion smoother. Therefore, there is no apparent difference for MPJAE.

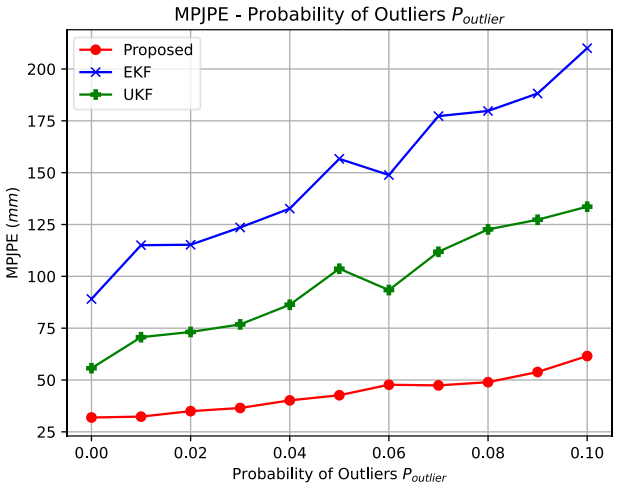
In Figure 6.14, there are three joint state estimators. One is the proposed estimator. Another one is extended Kalman filter (EKF). The other one is unscented Kalman filter (UKF). Since the EKF and the UKF don't have the ability to handle the missing data, they would replace the missing data with the predicted observation as Equation (6.29):

$$z_{l,t} = \begin{cases} \hat{z}_{l,t|t-1}, & if\ z_{l,t}\ missed \\ z_{l,t}, & else \end{cases}, \qquad \forall l = 1, \ldots, m \tag{6.29}$$

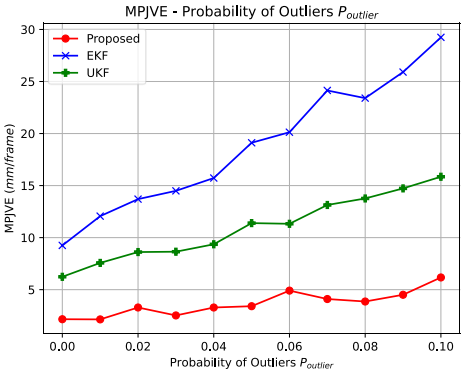
where $z_{l,t}$ is the l-th element in $\boldsymbol{z}_t \in \mathbb{R}^m$ and $\hat{z}_{l,t|t-1}$ is the l-th element in $\hat{\boldsymbol{z}}_{t|t-1} \in \mathbb{R}^m$.

The performance errors of the three joint state estimators in different $P_{outlier}$ are shown in Figure 6.14. All the performance errors grow up when $P_{outlier}$ increases. Besides, the proposed one always performs the best among the three joint state estimators. UKF is the second, and EKF is the worst. In the comparison between UKF and EKF, since the skeleton model is a highly nonlinear system. The sampling method adopted by UKF would perform better than the linearization method of partial derivatives adopted by EKF.

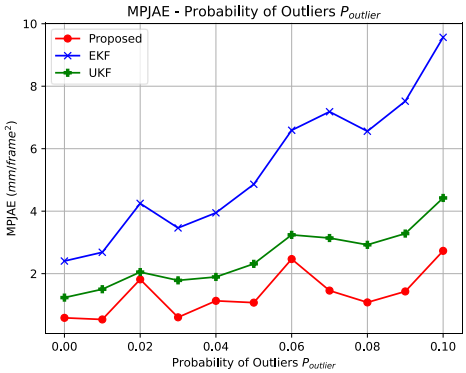
doi:10.6342/NTU202202468



(a)



(b)



(c)

Figure 6.14: The performance with different joint state estimators in different probability of outliers $P_{outlier}$
(a) Position Error (b) Velocity Error (c) Acceleration Error

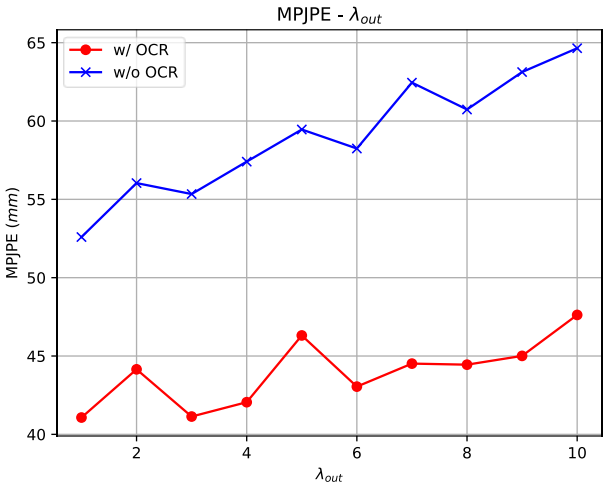
doi:10.6342/NTU202202468

2) Different expected outlier interval duration λ_{out}

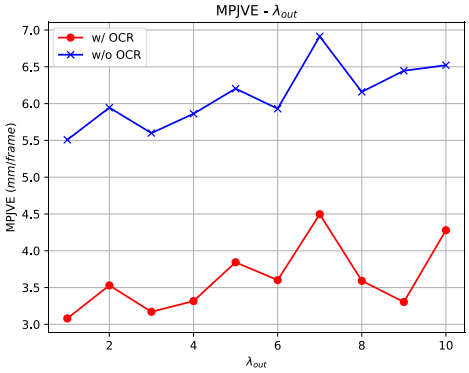
In this part, the expected outlier interval duration λ_{out} will the independent variable. As shown in Figure 6.15, MPJPE and MPJVE increase slightly as λ_{out} grows. Since $P_{outlier}$ remains the same, the expected ratio of outliers is still the same. However, when λ_{out} grows, the outliers will trend to gather as fewer and longer outlier intervals. With longer outlier intervals, the state covariance $\boldsymbol{P}_t$ and the bias of the predicted state $\hat{\boldsymbol{x}}_{t|t-\tau}$ in the proposed method will become higher since there is a longer time without measurement update. For the joint state estimator without OCR, the longer outlier intervals are more influential than the shorter but more outlier intervals as shown in Figure 6.15.

In Figure 6.15 (c), the influence of λ_{out} looks subtle. However, when $\lambda_{out} = 7,10$, there are two obvious peaks for the joint state estimator with OCR. It seems caused by other factors unrelative to λ_{out}. 10-time of re-test is not enough to wipe out the factors.

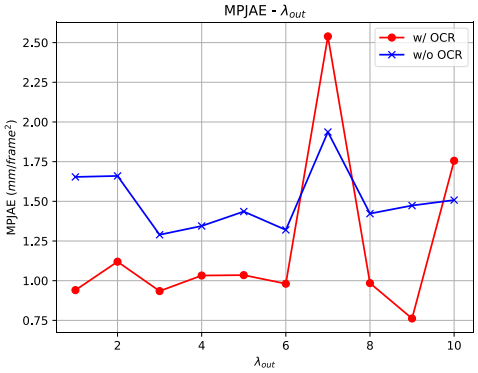
doi:10.6342/NTU202202468



(a)



(b)

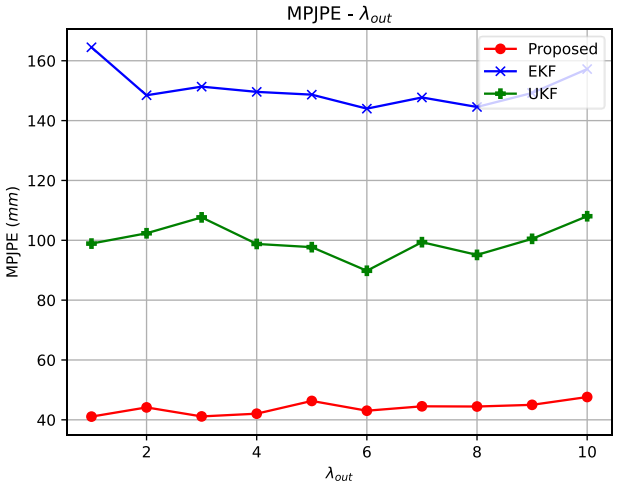


(c)

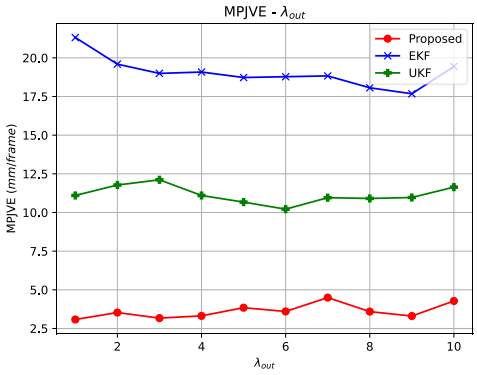
Figure 6.15: The performance with or without OCR (outlier component rejection) in different expected outlier interval duration λ_{out}
(a) Position Error (b) Velocity Error (c) Acceleration Error

doi:10.6342/NTU202202468

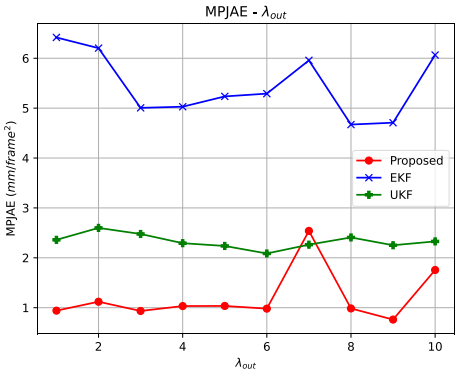
Compared with EKF and UKF, the proposed method still prevails with different λ_{out}. In Figure 6.16, we can notice that both EKF and UKF seem not to have an obvious trend in λ_{out}.



(a)



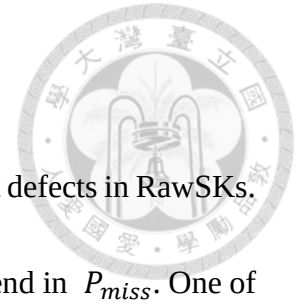
(b)



(c)

Figure 6.16: The performance with different joint state estimators in different expected outlier interval duration λ_{out}
(a) Position Error (b) Velocity Error (c) Acceleration Error

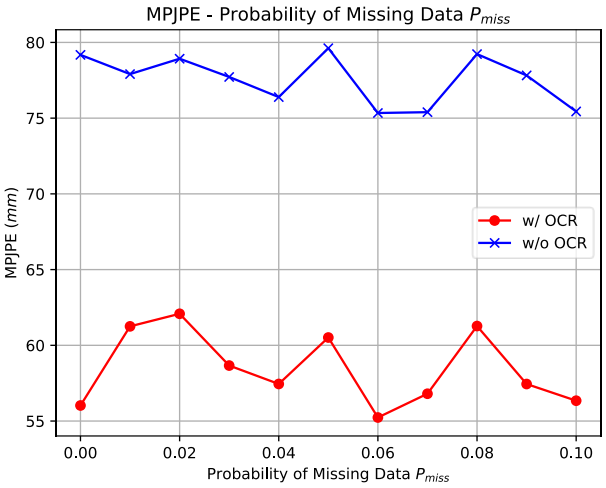
doi:10.6342/NTU202202468



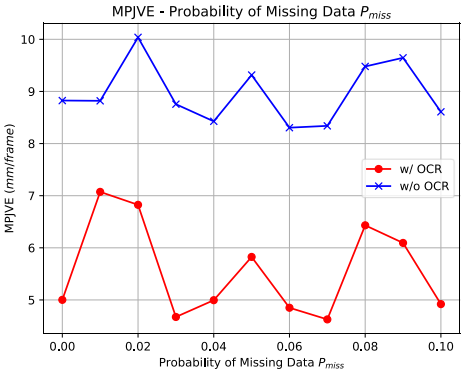
3) Different probability of missing data P_{miss}

Besides the outliers, the missing data is also one of the important defects in RawSKs. However, with the observation in Figure 6.17, there is no obvious trend in P_{miss}. One of the inferences is that missing data are perfectly labeled outliers for the proposed method with OCR. Unlike the real outliers, they won't be regarded as the inliers and influence the estimation results. Therefore, there is no obvious trend for the proposed method with OCR in P_{miss}. For the proposed method without OCR, the missing data are directly replaced with the predicted state as Equation (6.29). Thus, there is also no obvious trend for the proposed method without OCR.

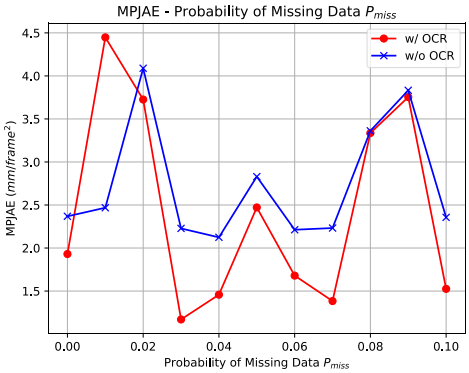
doi:10.6342/NTU202202468



(a)



(b)

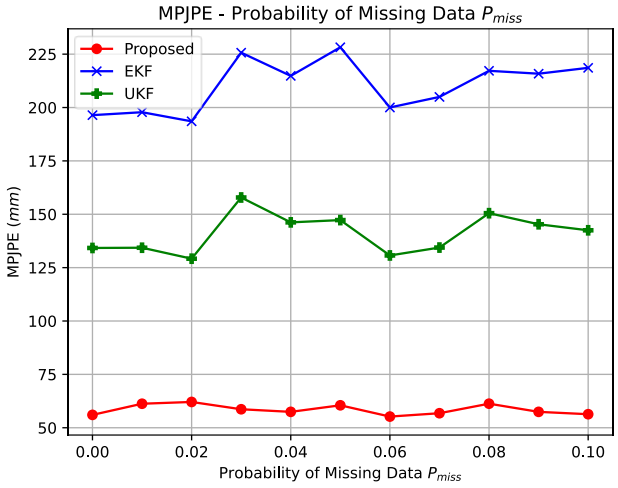


(c)

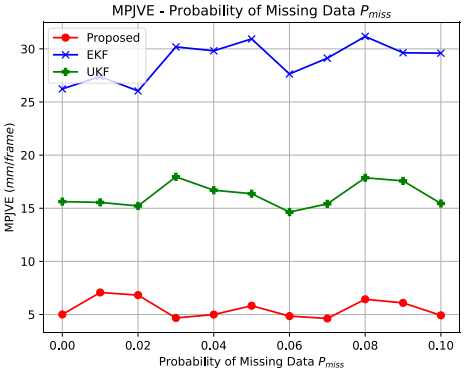
Figure 6.17: The performance with or without OCR (outlier component rejection) in different probability of missing data P_{miss}
(a) Position Error (b) Velocity Error (c) Acceleration Error

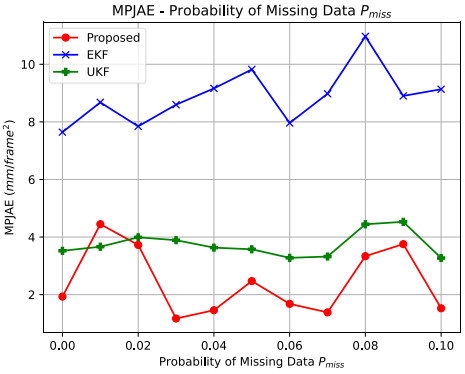
doi:10.6342/NTU202202468

The results are similar in Figure 6.18. There is no obvious trend in P_{miss} for the three joint state estimators. The proposed method prevails over the others in most cases.



(a)





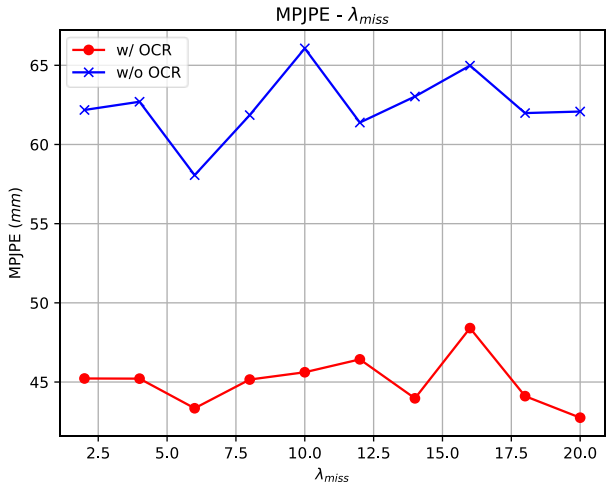
(b) (c)

Figure 6.18: The performance with different joint state estimators in different probability of missing data P_{miss}
(a) Position Error (b) Velocity Error (c) Acceleration Error

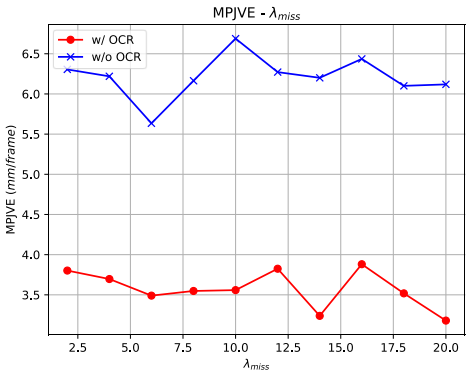
doi:10.6342/NTU202202468

4) Different expected missing interval duration λ_{miss}

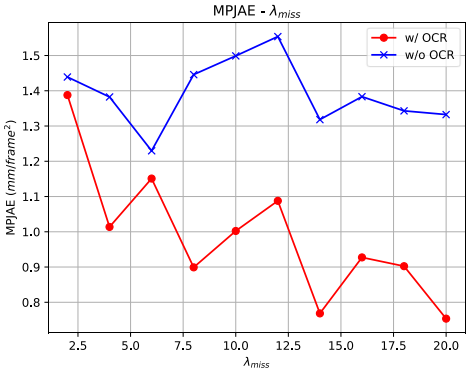
The last part is for λ_{miss}. As the results in part 3), there is no obvious trend in Figure 6.19 and Figure 6.20. Despite that there is a go-down trend in MPJAE for the proposed method with OCR, it seems caused by the fined smoothing results and the few sample times.



(a)



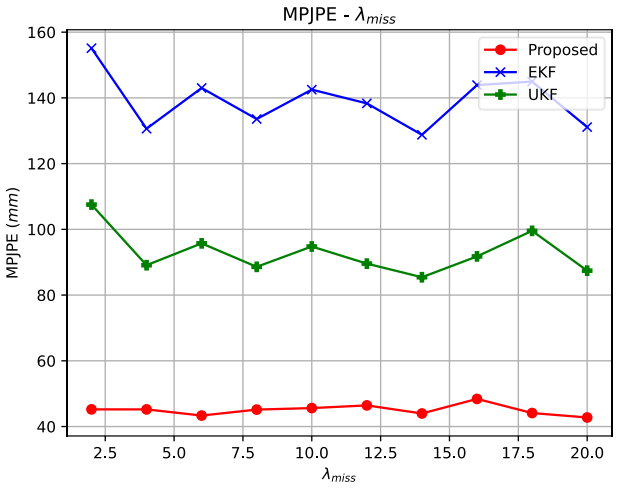
(b)



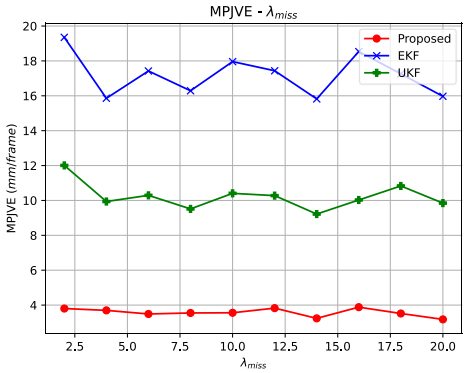
(c)

Figure 6.19: The performance with or without OCR (outlier component rejection) in different expected missing interval duration λ_{miss}
(a) Position Error (b) Velocity Error (c) Acceleration Error

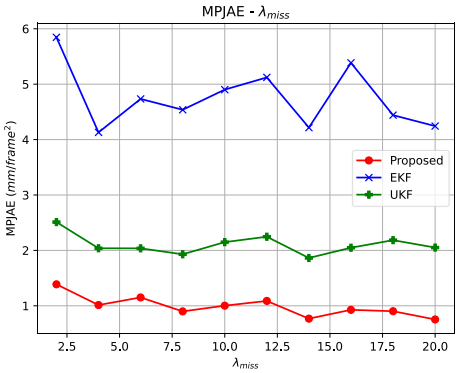
doi:10.6342/NTU202202468



(a)



(b)



(c)

Figure 6.20: The performance with different joint state estimators in different expected missing interval duration λ_{miss}
(a) Position Error (b) Velocity Error (c) Acceleration Error

In summary, the proposed method shows the best performance in different $P_{outlier}$ and λ_{out} compared with EKF and UKF. The effectiveness of outlier component rejection is also verified. When $P_{outlier}$ grows higher, the improvement of the OCR mechanism is more obvious.

For the missing data, there is no apparent trend no matter for P_{miss} or for λ_{miss}.

doi:10.6342/NTU202202468

6.4.3 Effectiveness of Iterative LQR Motion Smoother

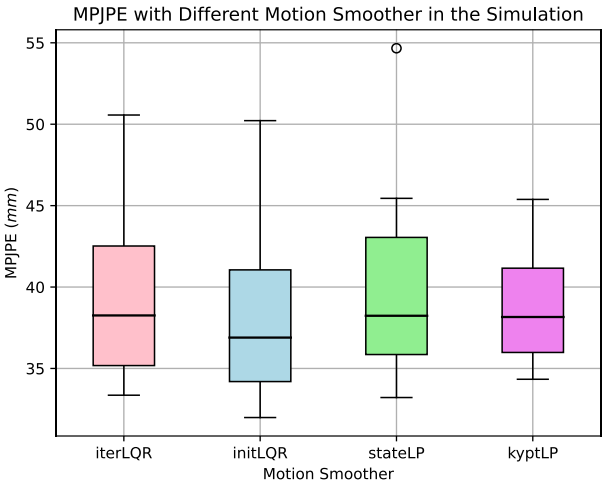
In this section, we will compare the proposed motion smoother -- iterative LQR tracking (iterLQR) with other motion smoother: the proposed initial LQR tracking (initLQR), state low pass filter (stateLP), and keypoint low pass filter (kyptLP).

The initLQR is the initialization part of iterLQR mentioned in Equation (5.87). the stateLP is to filter the joint state with a low pass filter directly. Last, the kyptLP is to filter the keypoint positions of KF-SK with a low pass filter.

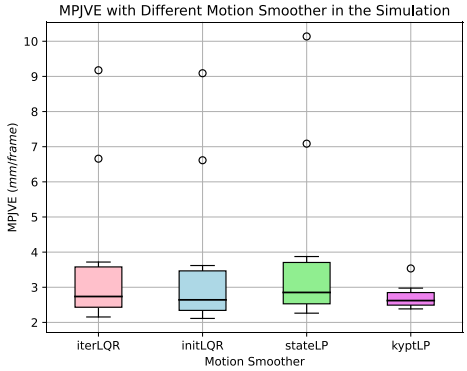
The low pass filter here is composed of the well-known Butterworth filter with fifth order and the zero-phase digital filtering described in [59: Gustafsson 1996] to avoid the phase delay in low pass filters. The cutoff frequency for the Butterworth filter is set as 0.025 times the sampled frequency (FPS).

The random properties of RawSK are set the same as the one mentioned in Section 6.3.2 while repeating the test with 10 different samples. The overall performance is shown in Figure 6.21. Among the four motion smoothers, initLQR has the best performance in MPJPE and MPJVE for this motion. However, the statistical dispersions of the LQR-based methods are higher than the low-pass-filter-based methods. Besides, the differences among the motion smoothers are relatively small to their dispersions.

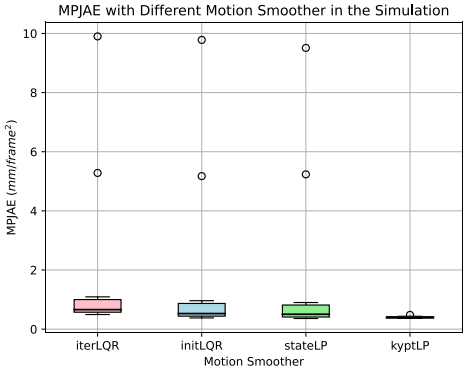
doi:10.6342/NTU202202468



(a)



(b)



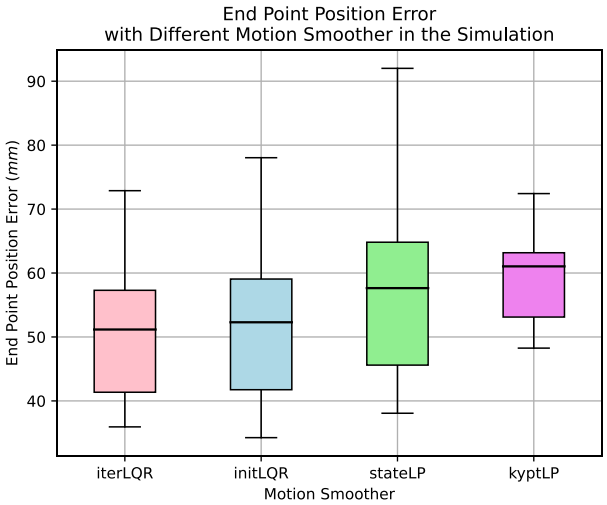
(c)

Figure 6.21: The performance with different motion smoother in the simulation (a) Position Error (b) Velocity Error (c) Acceleration Error

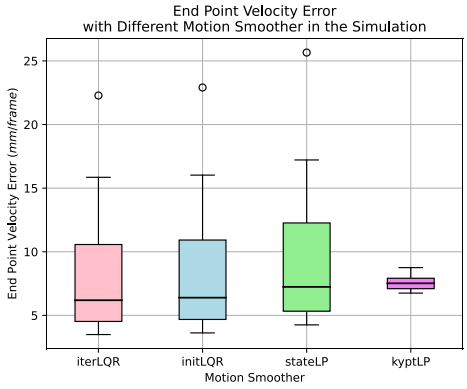
As the design concept of iterLQR is to improve the end point position error caused by the joint coordination problem, the performance of the endpoint $\boldsymbol{p}_5$ of SimSK is shown in Figure 6.22. Statistically, the proposed iterLQR has the minimum median values for position and velocity error. For the acceleration, the iterLQR, initLQR and stateLP have similar performance while the kyptLP has much less dispersion. It's because the kyptLP is the only one smoother that directly smooths the keypoint positions without

doi:10.6342/NTU202202468

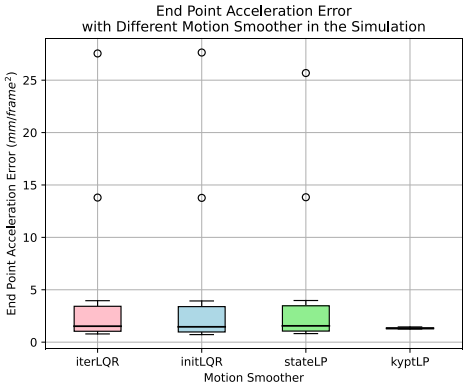
considering the joint state. Since the nonlinear system $f(\cdot)$ and $h(\cdot)$ of a human skeleton is not fully observable, the joint state may vary hugely even when the keypoint positions in RawSKs are similar. Therefore, the kyptLP without considering the joint state has a much smaller variation compared with the others.



(a)



(b)



(c)

Figure 6.22: The end point performance with different motion smoother in the simulation
(a) Position Error (b) Velocity Error (c) Acceleration Error

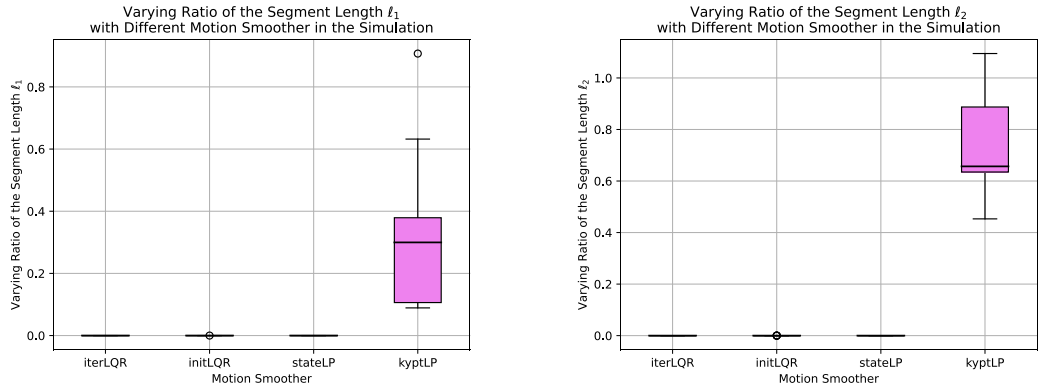
doi:10.6342/NTU202202468

Nevertheless, also caused by the direct keypoint smoothing, the skeleton kinematic model is not considered for the kyptLP. The segment lengths smoothed by the kyptLP would not be constant. To evaluate the variation of segment lengths, we define a metric-varying ratio $r_v(\ell)$ as:

$$r_v(\ell) = \frac{\ell_{max} - \ell_{min}}{\mu_\ell} \tag{6.30}$$

where ℓ is the varying segment length, ℓ_{max} is the maximum value of ℓ, ℓ_{min} is the minimum value of ℓ, and μ_ℓ is the mean value of ℓ.

The varying ratios of the two segment length in SimSK are plotted in Figure 6.23. The state-filtering-based motion smoothers (iterLQR, initLQR, stateLP) have zero-value varying ratios due to the skeleton kinematic model. However, for kyptLP, its varying ratios are considerable.



(a) (b)

Figure 6.23: Segment length varying ratio in the simulation
(a) length of upper section ℓ_1 (b) length of lower section ℓ_2

doi:10.6342/NTU202202468

To demonstrate the advantage of the proposed iterLQR further, a special but common motion in real life is investigated. The motion is a fix-end-point motion illustrated in Figure 6.24. The relative pose of the skeleton remains the same as the tested motion in Section 6.3.2 while the end point $\boldsymbol{p}_5$ is fixed at the origin. Analogizing to a human skeleton, the motion is common when one of the feet is fixed on the ground. The other keypoints can do the relative motion while the endpoint is fixed.

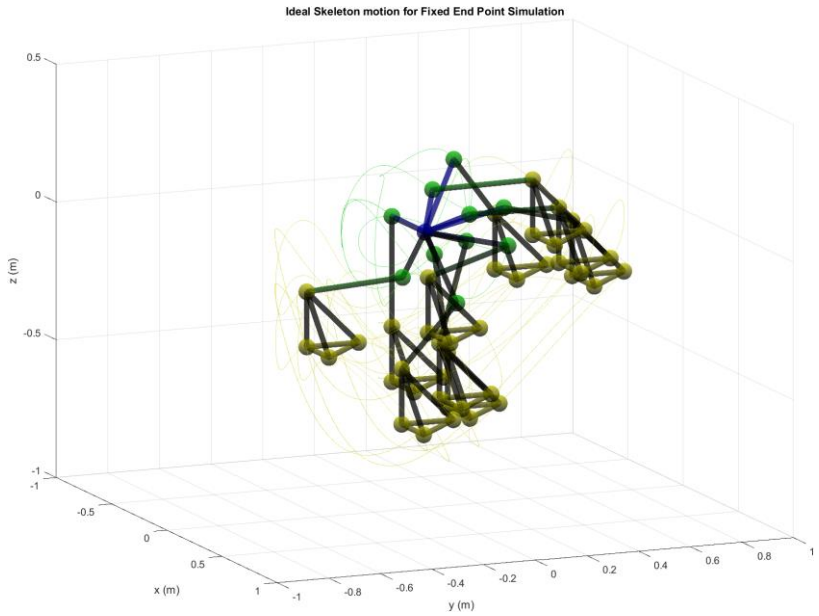
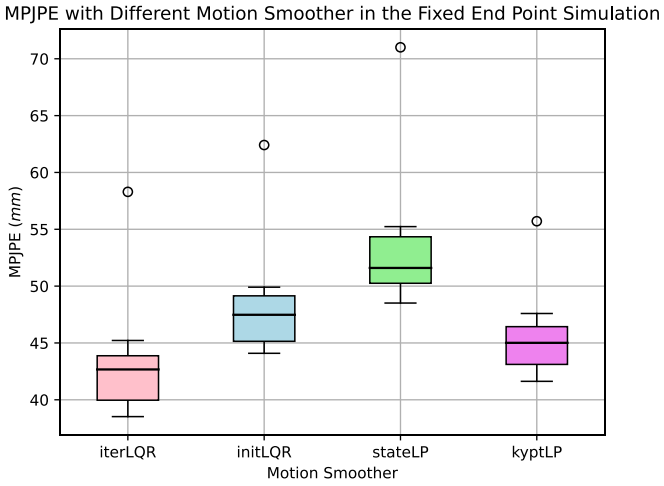


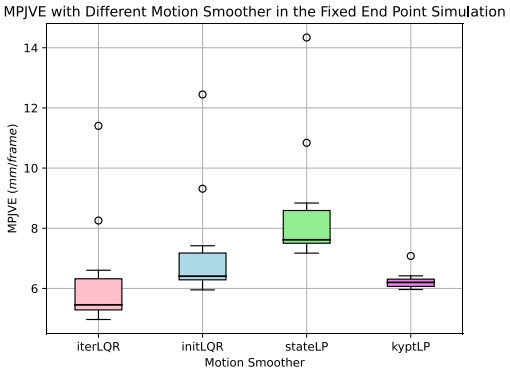
Figure 6.24: The ideal motion (gtSK) of SimSK in fixed-end-point simulation

doi:10.6342/NTU202202468

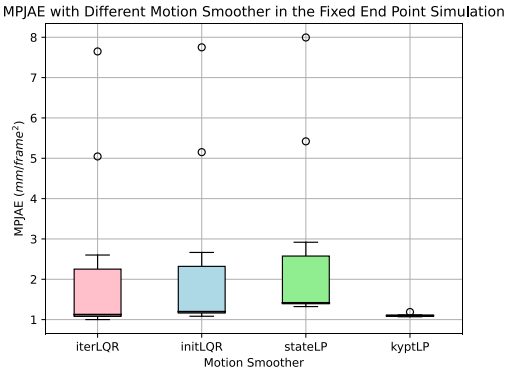
In the fixed-end-point motion, the average performances with different motion smoother are shown in Figure 6.25. The average errors of iterLQR are lower than others' in MPJPE and MPJVE.



(a)



(b)

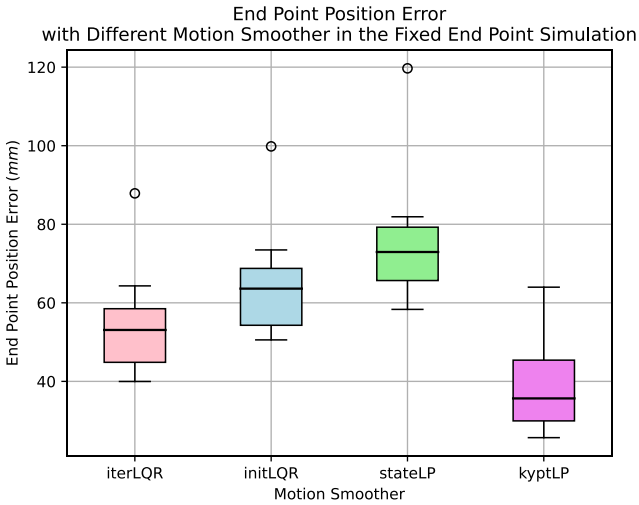


(c)

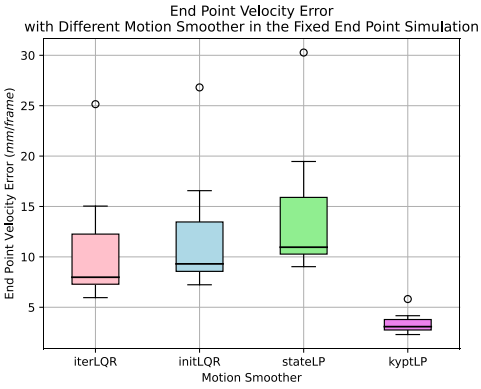
Figure 6.25: The performance with different motion smoother in fixed-end-point simulation
(a) Position Error (b) Velocity Error (c) Acceleration Error

doi:10.6342/NTU202202468

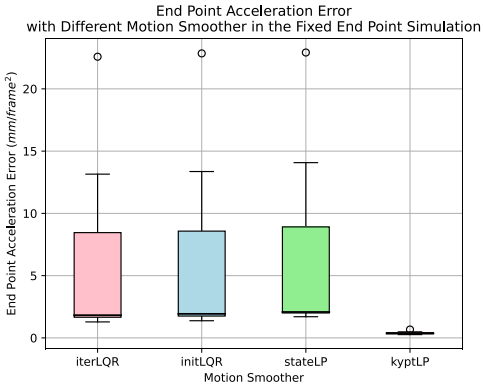
For the endpoint performance in Figure 6.26, the iterLQR also performs the best among the state-filtering-based motion smoother (iterLQR, initLQR, stateLP). For the kyptLP, since the endpoint position is fixed, the endpoint performance is much better than the others.



(a)



(b)

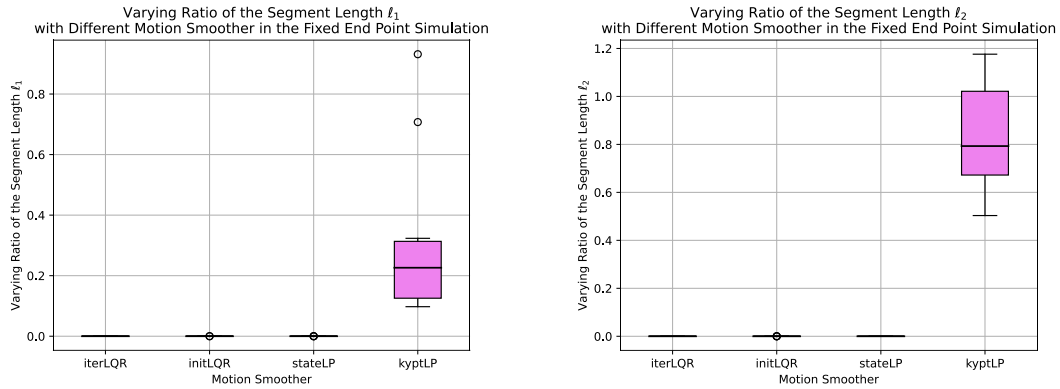


(c)

Figure 6.26: The end point performance with different motion smoother in fixed-end-point simulation
(a) Position Error (b) Velocity Error (c) Acceleration Error

doi:10.6342/NTU202202468

However, since the kyptLP doesn't consider the skeleton kinematic model, the varying ratios of the segment lengths with the kyptLP are also great in this motion.



(a) (b)

Figure 6.27: Segment length varying ratio in fixed-end-point simulation
(a) length of upper section ℓ_1 (b) length of lower section ℓ_2

doi:10.6342/NTU202202468

6.4.4 Influence of the Motion Speed

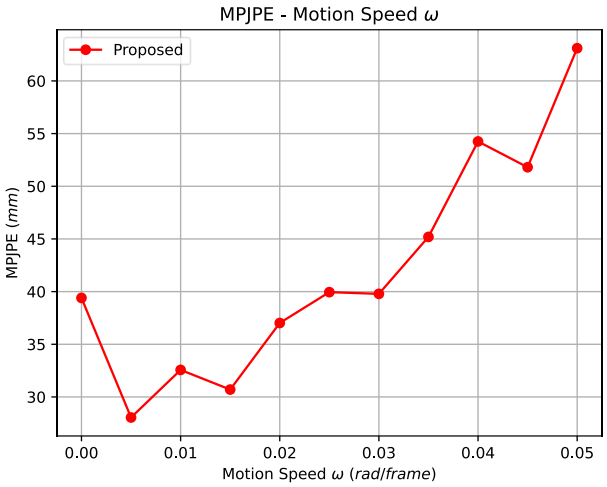
In the last section of the simulation, the influence of the motion speed will be the independent variable.

As the ideal motion mentioned in Equation (6.10)-(6.11), the motion speed would be affected by ω whose default value is 0.03. With higher ω, the motion speed becomes higher. Then, the performance with different ω is shown in Figure 6.28.

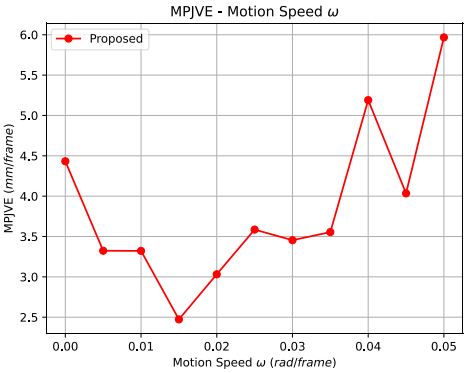
As ω grows, both MPJPE and MPJVE have apparent increasing trends. The reason caused from that the state transition function $f(\cdot)$ for the joint state estimator described in Equation (5.43) is the constant velocity model. When the joint acceleration input is unknown, the estimation error will increase as the joint acceleration becomes more and more unignorable.

On the other hand, increasing the sampling frequency also means reducing the motion speed between the frames. Therefore, it's potential to enhance the performance of the real-world experiment by increasing the FPS of the capture system.

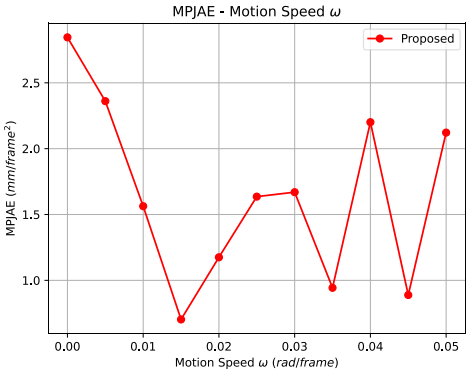
doi:10.6342/NTU202202468



(a)



(b)



(c)

Figure 6.28: The end point performance with different motion speeds ω
(a) Position Error (b) Velocity Error (c) Acceleration Error

doi:10.6342/NTU202202468

6.5 Experiment Setups

To evaluate the estimation error, a commercial optoelectronic motion capture system, VICON, is utilized as the ground truth provider in this experiment. The VICON markers are attached to the skin and leggings of the target to acquire the ground truth of the keypoints positions.

However, since we can't attach the VICON markers on the target's face which would affect the detection of AlphaPose, the ground truth positions of head keypoints are not accessible. Therefore, in the experiment, we only compare the 12 keypoints on the body.

The details of the multi-view system construction will be presented in Section 6.5.1. The tested motions in the experiment are shown in Section 6.5.2. Then, the parameter settings for the proposed motion estimation method will be listed in Section 6.5.3.

doi:10.6342/NTU202202468

6.5.1 Multi-view system setups

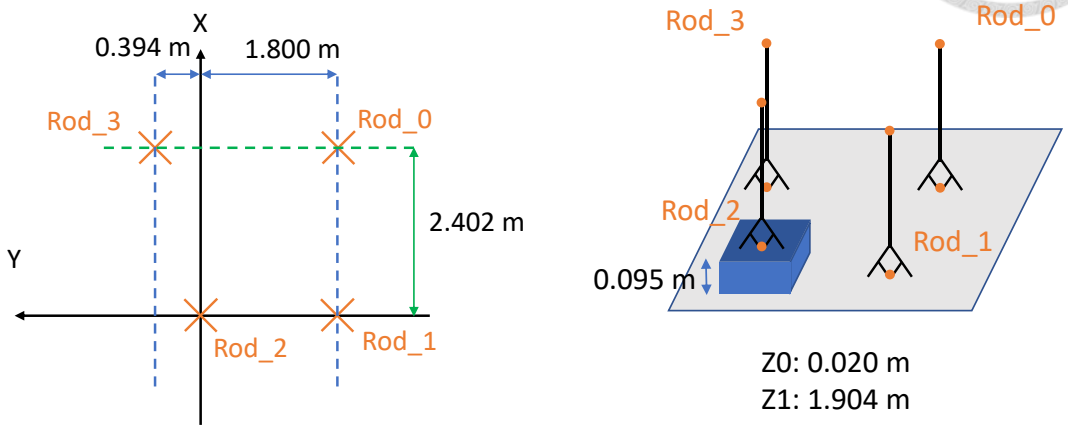
To fulfill the proposed vision-based motion estimation method, the multi-view system mentioned in Chapter 4 should be installed. As Section 4.1 said, the proposed multi-view system is composed of four synchronized cameras capturing the target videos from different views. The positions of these four cameras in the experiment are shown in Figure 6.29 marked with red circles. In the experiment, the multi-view system and VICON will record simultaneously.

Figure 6.29: The camera positions set for VICON evaluation

To define the world coordinate $\{world\}$, the calibration process mentioned in Section 4.2 is applied. There are four rod positions with eight reference points illustrated in Figure 6.30 for the extrinsic calibration. After the calibration, the world coordinate $\{world\}$ for the experiment is defined by the positions of the reference points. Besides, to transform the ground truth position of the keypoints to $\{world\}$, the transformation

doi:10.6342/NTU202202468

matrix from VICON to $\{world\}$ are also calculated with the VICON markers attached to the reference points.



(a) (b)

Figure 6.30: The positions of reference points for VICON evaluation
(a) XY-positions (b) Z-positions

Going through the calibration process, the camera parameters for the multi-view system are estimated as Table 6.6 and Table 6.7 show.

doi:10.6342/NTU202202468

Table 6.6

Intrinsic and Extrinsic Camera Parameters for Multi-view System in VICON Evaluation

	Intrinsic Parameter $\boldsymbol{K}$	Extrinsic Parameter ${}^{world}_{cam}\boldsymbol{T}$
Cam0	$\begin{bmatrix} 513.28 & 0 & 363.39 \\ 0 & 515.15 & 301.41 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.620 & 0.784 & 0.015 & -0.164 \\ 0.085 & -0.049 & -0.995 & 0.595 \\ -0.780 & 0.619 & -0.097 & 5.192 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Cam1	$\begin{bmatrix} 511.26 & 0 & 336.75 \\ 0 & 514.04 & 261.23 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0.373 & -0.928 & 0.004 & -0.999 \\ -0.085 & -0.038 & -0.996 & 1.015 \\ 0.924 & 0.371 & -0.093 & 3.137 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Cam2	$\begin{bmatrix} 510.11 & 0 & 378.04 \\ 0 & 511.65 & 275.79 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.241 & -0.970 & 0.015 & -0.110 \\ 0.015 & -0.019 & -1.000 & 1.092 \\ 0.970 & -0.241 & 0.019 & 2.245 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Cam3	$\begin{bmatrix} 511.25 & 0 & 357.65 \\ 0 & 513.45 & 240.22 \\ 0 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -0.764 & 0.645 & 0.014 & 1.600 \\ 0.190 & 0.246 & -0.950 & 1.088 \\ -0.616 & -0.724 & -0.310 & 3.283 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

Table 6.7

Distortion Coefficients of Cameras for Multi-view System in VICON Evaluation

	Distortion Coefficients $[k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3]$
Cam0	$[-0.180 \quad 0.262 \quad -0.001 \quad -0.002 \quad -0.286]$
Cam1	$[-0.164 \quad 0.194 \quad -0.002 \quad -0.003 \quad -0.176]$
Cam2	$[-0.144 \quad 0.046 \quad 0.001 \quad -0.003 \quad 0.077]$
Cam3	$[-0.132 \quad -0.005 \quad 0.000 \quad -0.002 \quad 0.167]$

doi:10.6342/NTU202202468

With the calibrated extrinsic parameters, the positions and the orientations of the cameras can be illustrated in Figure 6.31. The axis marked with the black point for each camera shows its facing direction (z-axis). As Figure 6.31 shown, all the cameras approximately face the origin of $\{world\}$, which is around the position of the target. The camera positions are also similar to the relative positions shown in Figure 6.29.

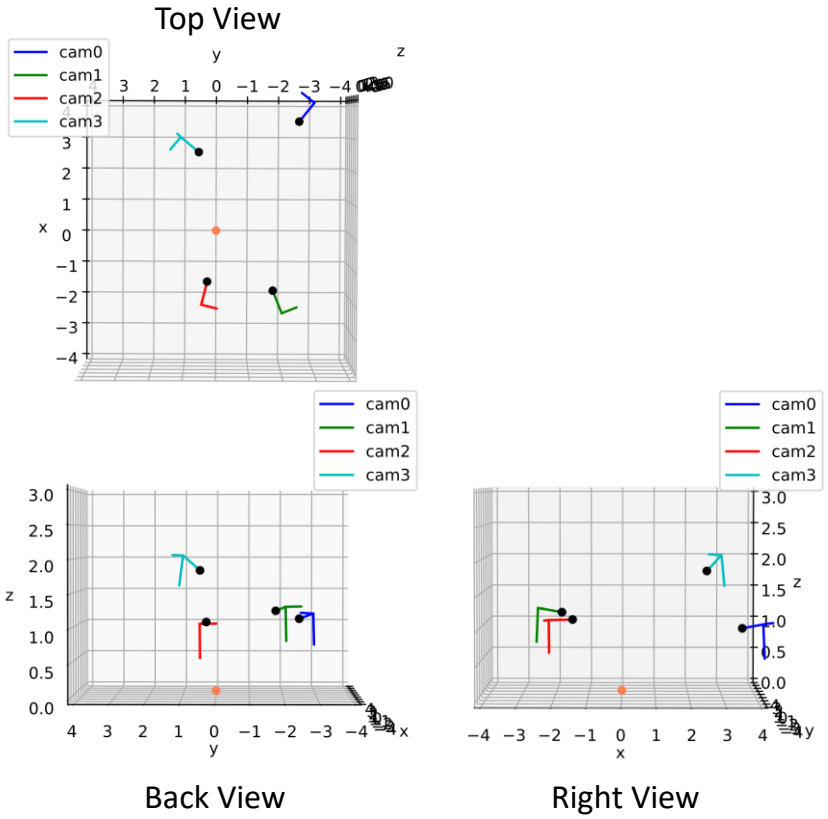


Figure 6.31: The camera positions calculated with the calibrated extrinsic parameters (the axis with black points are the z-axis of cameras)

In addition, to make it easier to find the correspondence between the estimated results and the ground truth provided by VICON, the recording FPS of the multi-view system is the same as VICON's FPS at 120 Hz.

doi:10.6342/NTU202202468

6.5.2 Tested Motions

In this experiment, we present three actions with the same subject (target person). These three actions are two common motions during baseball games (hitting and pitching) and one motion (punching) similar to pitching but with different trajectories in right arms.

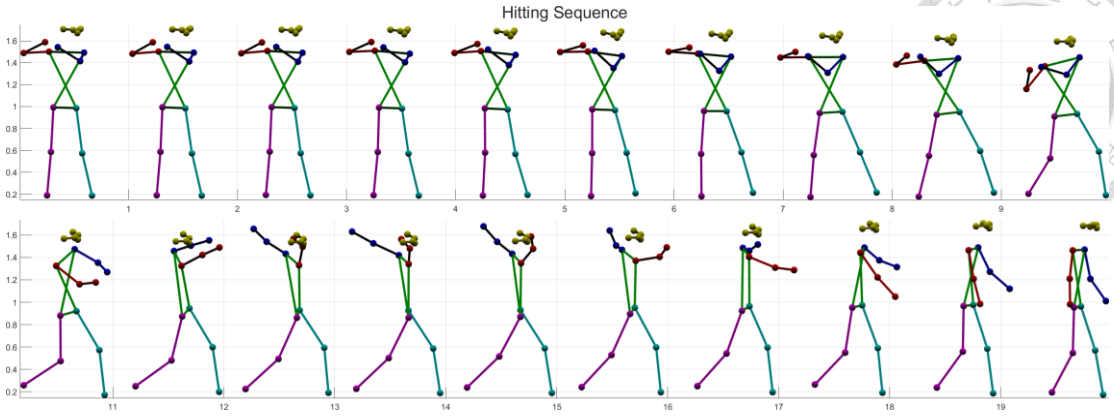
For each action, there are five cases to evaluate the reproducibility of the proposed method. All the cases are listed in Table 6.8 and labeled with the case index from 0 to 14.

Table 6.8
Testing Cases with VICON Evaluation

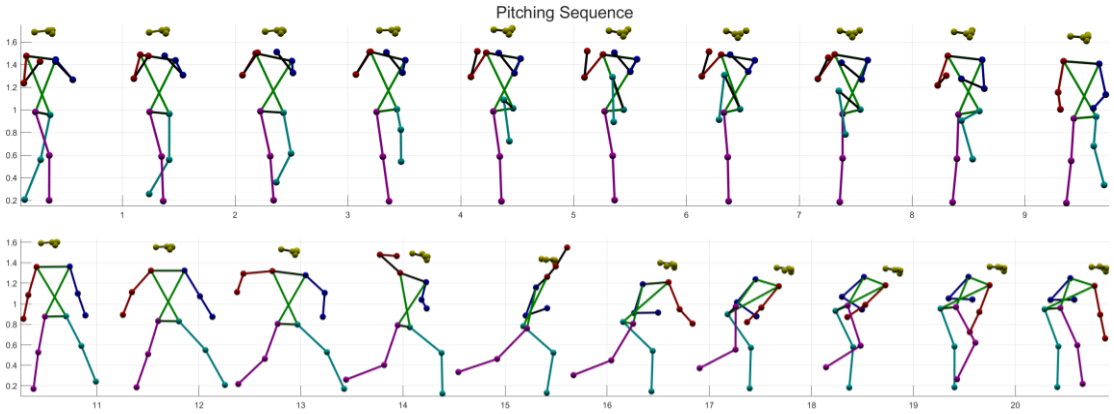
Subject	Action	Case Name	Case Index
S1	Hitting	Hitting_01	0
		Hitting_02	1
		Hitting_03	2
		Hitting_04	3
		Hitting_05	4
	Pitching	Pitching_01	5
		Pitching_02	6
		Pitching_03	7
		Pitching_04	8
		Pitching_05	9
	Punching	Punching_01	10
		Punching_02	11
		Punching_03	12
		Punching_04	13
		Punching_05	14

The motion sequence of each action is illustrated in Figure 6.32. Hitting is acted as a batter hits the ball in a baseball game. Pitching is a motion of a pitcher throwing a ball. Then, punching is a motion similar to pitching, but the right arm is punching forward.

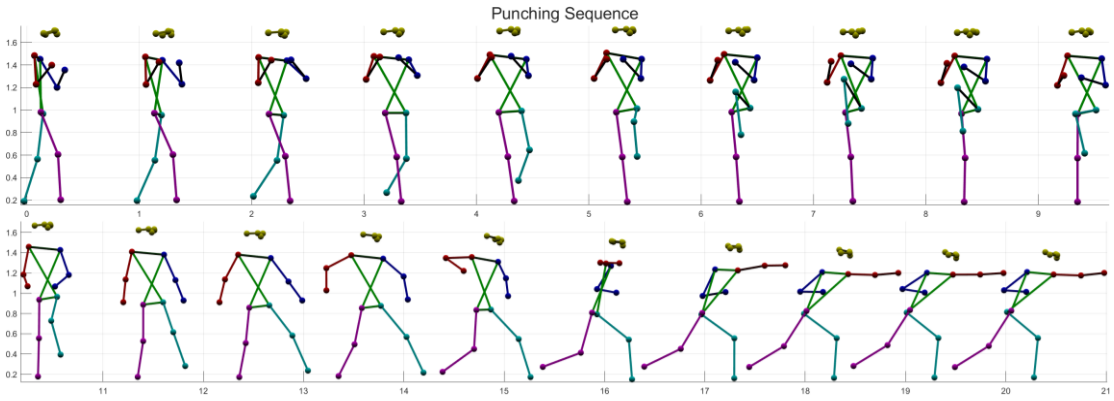
doi:10.6342/NTU202202468



(a)



(b)



(c)

Figure 6.32: The motion sequence of the testing actions in VICON evaluation plotted with estimated positions by proposed method
(a) Hitting (b) Pitching (c) Punching

doi:10.6342/NTU202202468

6.5.3 Parameter Settings

In the experiment, the parameters set for the proposed estimation method are shown in Table 6.9 and Table 6.10:

Table 6.9
The Parameter Settings for the Joint State Estimator in the Proposed Method in the Experiment

Joint State Estimator:		
Symbol	Value	Meaning
$\boldsymbol{P}_0$	$10^{-1} \cdot \boldsymbol{I}_{56}$	Initial state covariance
$\boldsymbol{Q}_{UKF}$	$\begin{bmatrix} \mathbf{0}_{22\times22} & \mathbf{0}_{22\times22} & \mathbf{0}_{22\times6} & \mathbf{0}_{22\times6} \\ \mathbf{0}_{22\times22} & 10^{-5} \cdot \boldsymbol{I}_{22} & \mathbf{0}_{22\times6} & \mathbf{0}_{22\times6} \\ \mathbf{0}_{6\times22} & \mathbf{0}_{6\times22} & \mathbf{0}_{6\times6} & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times22} & \mathbf{0}_{6\times22} & \mathbf{0}_{6\times6} & 10^{-5} \cdot \boldsymbol{I}_{6} \end{bmatrix}$	Process noise covariance
$\boldsymbol{R}_{UKF}$	$10^{-4} \cdot \boldsymbol{I}_{51}$	Measurement noise covariance
k_{out}	0.05	Outlier rejecting coefficient
$\boldsymbol{Q}_{in,N_{out}}$	$10^{10} \cdot \boldsymbol{I}_{56}$	State weights for inlier interpolation
$\boldsymbol{R}_{in}$	$\boldsymbol{I}_{28}$	Input weights for inlier interpolation
$\boldsymbol{x}_{lower}$	$-[\infty \cdot \mathbf{1}_{22} \quad 0.1 \cdot \mathbf{1}_{22} \quad \infty \cdot \mathbf{1}_{6} \quad \infty \cdot \mathbf{1}_{6}]^T$	State lower bound with KKT condition

doi:10.6342/NTU202202468

	$\boldsymbol{x}_{upper}$	$[\infty \cdot \mathbf{1}_{22} \quad 0.1 \cdot \mathbf{1}_{22} \quad \infty \cdot \mathbf{1}_{6} \quad \infty \cdot \mathbf{1}_{6}]^{T}$	State upper bound with KKT condition
	c_{damp}	-0.1	Damper coefficient for body force

doi:10.6342/NTU202202468

Table 6.10

The Parameter Settings for the Motion Smoother in the Proposed Method in the Experiment

Motion Smoother:			
	Symbol	Value	Meaning
	$\boldsymbol{Q}_{lqr1}$	$\boldsymbol{I}_{56}$	State weights for initial LQR tracking
	$\boldsymbol{R}_{lqr1}$	$10^3 \cdot \boldsymbol{I}_{28}$	Input weights for initial LQR tracking
	$\boldsymbol{Q}_{lqr2}$	$\begin{bmatrix} \boldsymbol{I}_{24} & \boldsymbol{0}_{24\times6} & \boldsymbol{0}_{24\times15} & \boldsymbol{0}_{24\times6} \\ \boldsymbol{0}_{6\times24} & 10 \cdot \boldsymbol{I}_6 & \boldsymbol{0}_{6\times15} & \boldsymbol{0}_{6\times6} \\ \boldsymbol{0}_{15\times24} & \boldsymbol{0}_{15\times6} & \boldsymbol{I}_{15} & \boldsymbol{0}_{15\times6} \\ \boldsymbol{0}_{6\times24} & \boldsymbol{0}_{6\times6} & \boldsymbol{0}_{6\times15} & 10 \cdot \boldsymbol{I}_6 \end{bmatrix}$	State weights for iterative LQR tracking
	$\boldsymbol{Q}_{lqr2,vel}$	$\begin{bmatrix} \boldsymbol{0}_{22\times22} & \boldsymbol{0}_{22\times22} & \boldsymbol{0}_{22\times6} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{22\times22} & 10^4 \cdot \boldsymbol{I}_{22} & \boldsymbol{0}_{22\times6} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times6} & \boldsymbol{0}_{6\times6} \\ \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times22} & \boldsymbol{0}_{6\times6} & 10^4 \cdot \boldsymbol{I}_6 \end{bmatrix}$	Velocity state weights for iterative LQR tracking
	$\boldsymbol{R}_{lqr2}$	$\begin{bmatrix} 10^2 \cdot \boldsymbol{I}_{22} & \boldsymbol{0}_{22\times6} \\ \boldsymbol{0}_{6\times22} & 10^3 \cdot \boldsymbol{I}_6 \end{bmatrix}$	Input weights for iterative LQR tracking
	N_{iter}	3	Number of iterations for iterative LQR tracking

doi:10.6342/NTU202202468

6.6 Experiment Results and Analysis

6.6.1 Result of 3D Reconstruction

In the simulation, the RawSKs have already been given by the ideal motions with additional noise. However, in the experiment, we need to capture the RawSKs from the real human motions to form the RawSKs. With the 3D reconstruction method mentioned in Chapter 4, the RawSKs are built as Figure 6.33 shown.

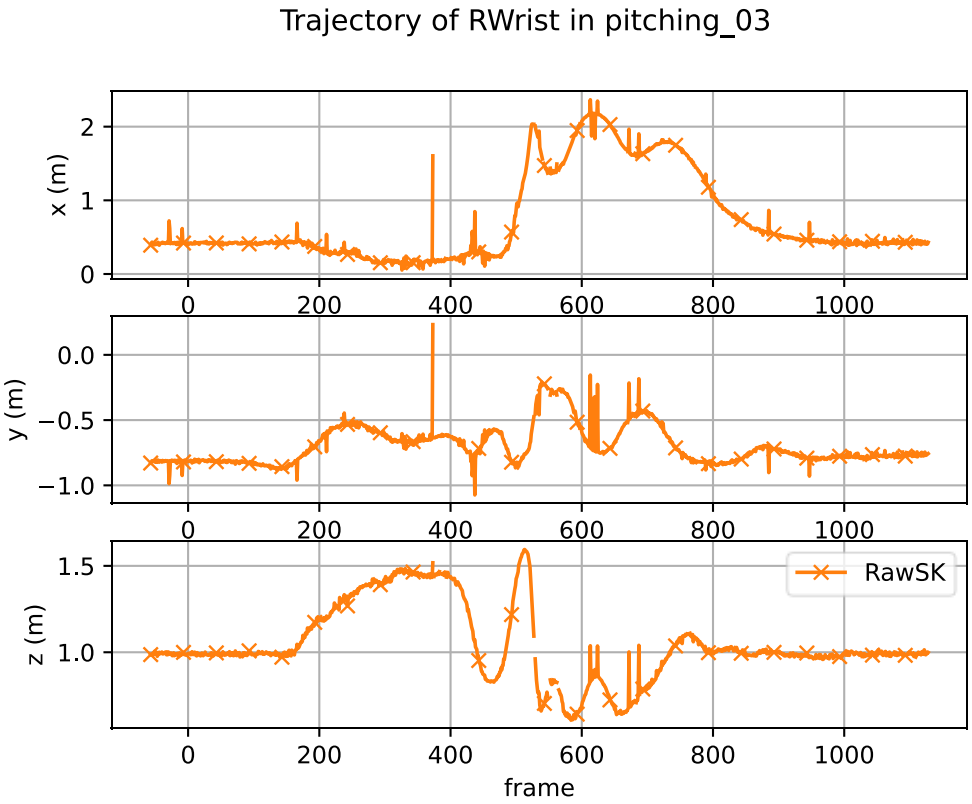


Figure 6.33: Example of keypoint trajectory in RawSKs in experiment

As we expected, the RawSKs are pretty **rough** and have some detects influencing the estimation results potentially, such as **outliers** and **missed data**. To quantify the defects of RawSK as in the simulation, the outlier ratio and average outlier interval

doi:10.6342/NTU202202468

duration will be used to analogize the outlier probability $P_{outlier}$ and the expected outlier interval duration λ_{out} in the simulation. For the missing data, the missing data ratio and the average missing interval duration are used to analogize the probability of missing data P_{miss} and the expected missing interval duration λ_{miss}.

For the missing data, it's easy to calculate their properties in RawSKs since they are labeled with NaNs directly. However, the outliers don't have a natural definition to mark. For that reason, we simply define the outliers in RawSK as:

$$Outlier: \sqrt{\left(\boldsymbol{p}_{i,raw}(t)-\boldsymbol{p}_{i,gt}(t)\right)^{T}\boldsymbol{S}_{\boldsymbol{p},i}^{-1}\left(\boldsymbol{p}_{i,raw}(t)-\boldsymbol{p}_{i,gt}(t)\right)} \geq k_{out,raw} \qquad (6.31)$$

where $\boldsymbol{p}_{i,raw}(t)$ is the keypoint position for the i-th keypoint in frame t in RawSK, $\boldsymbol{p}_{i,gt}(t)$ is the keypoint position for the i-th keypoint in frame t in ground truth skeleton, and $\boldsymbol{S}_{\boldsymbol{p},i}$ is the covariance of $\boldsymbol{p}_{i,raw}(t)-\boldsymbol{p}_{i,gt}(t)$ for all t in the i-th keypoint.

While plotting the position error vectors $\boldsymbol{p}_{i,raw}(t)-\boldsymbol{p}_{i,gt}(t)$ for all t as shown in Figure 6.34, there are some points far from the group around the origin. To pick those points out as outliers, we use the outlier definition as Equation (6.31) imitating the Mahalanobis distance while setting $k_{out,raw}=5$. Then, the outlier vectors are labeled with red color, and the inliers are labeled with blue color in Figure 6.34.

doi:10.6342/NTU202202468

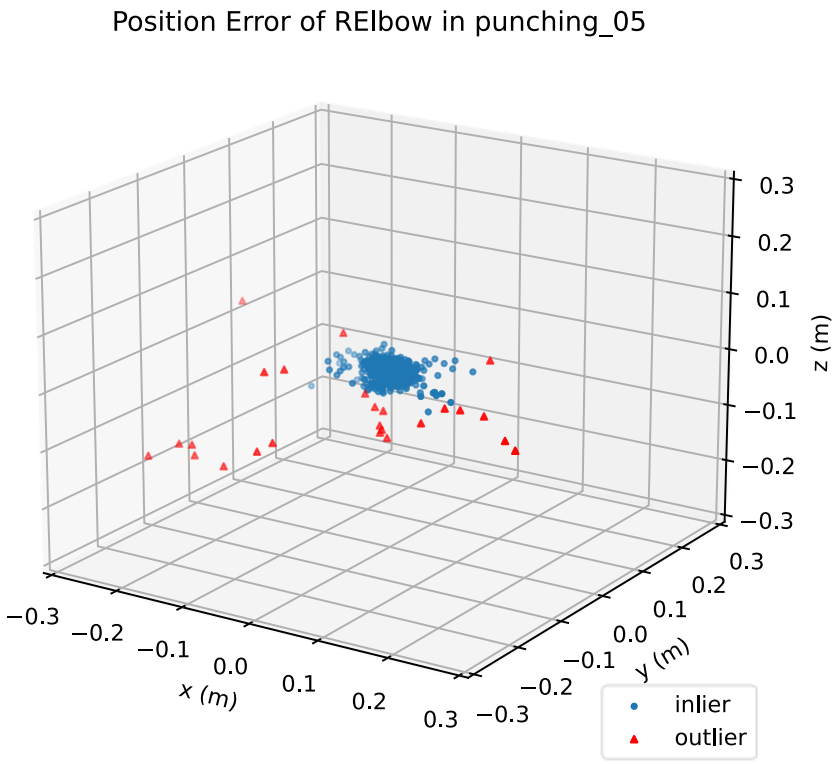


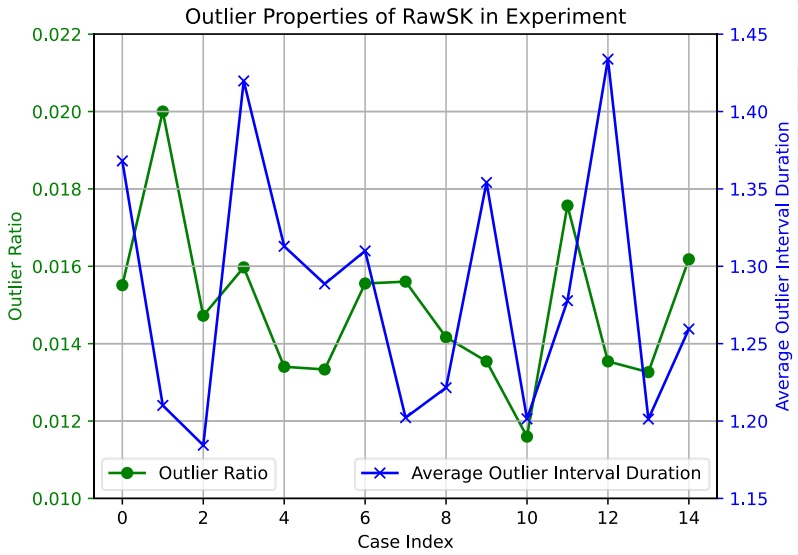
Figure 6.34: Outlier Labeled result for RElbow of RawSK in punching_05

With the labeled outliers and missing data, we can calculate the outlier and missing data properties as shown in Figure 6.35.

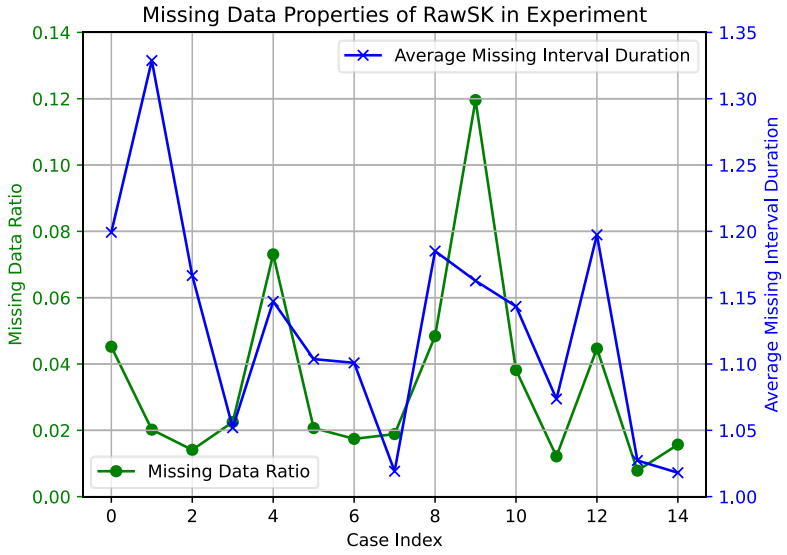
For the AlphaPose, proposed multi-view system and 3D reconstruction method, the outlier ratios are between 1.0% to 2.2%, and the average outlier interval durations are between 1.15 to 1.45 frames.

For missing data, the missing data ratios are between 0% to 14%. Finally, the average missing interval durations are between 1.00 to 1.35 frames.

doi:10.6342/NTU202202468



(a)



(b)

Figure 6.35: Outlier and missing data properties of RawSK
(a) Outlier properties (b) Missing data properties

doi:10.6342/NTU202202468

Besides the properties of the potential influencing factors, the position errors MPJPE of the RawSKs in the experiment are illustrated in Figure 6.36.

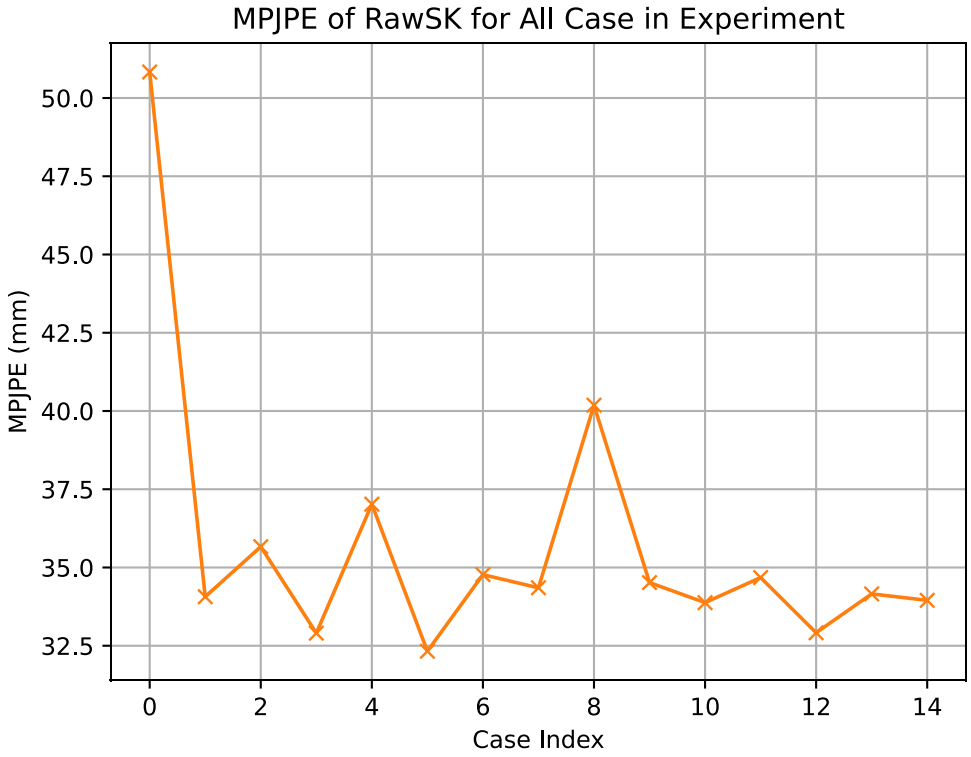


Figure 6.36: MPJPE of RawSKs in experiment

The overall position error of RawSKs is about 30mm to 40mm except Hitting_01. In Figure 6.37, we can find that AlphaPose is bad at arm keypoint detection with the bat-lifting pose. Particularly, the keypoint of LWrist is marked out of the arm at the frame. In Hitting_01, the target remained the bat-lifting pose for a much longer time than in the other hitting cases. Therefore, the MPJPE is especially great in Hitting_01.

doi:10.6342/NTU202202468

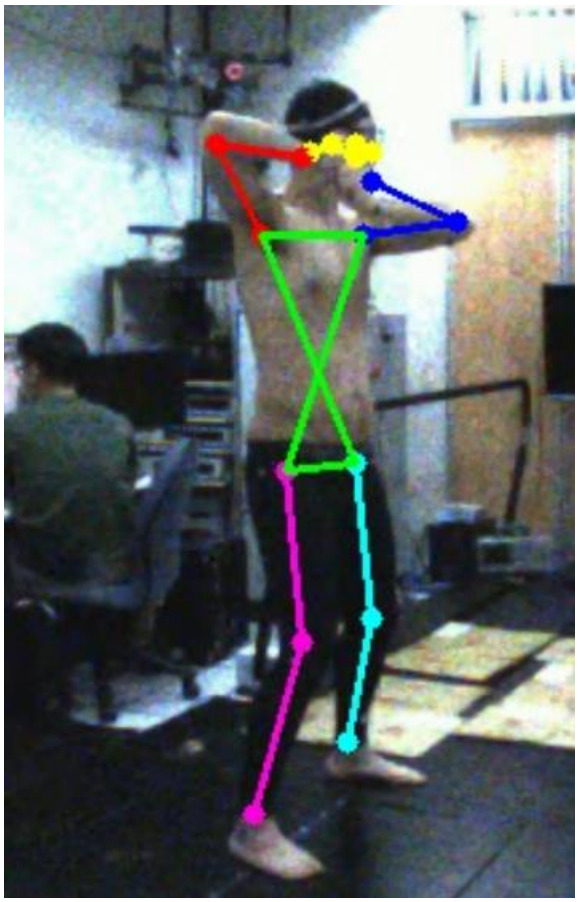


Figure 6.37: AlphaPose detection result for Cam1 in Hitting_01

doi:10.6342/NTU202202468

6.6.2 Performance Analysis of Motion Estimation

After the 3D reconstruction for the RawSKs, the 3D skeleton modification method presented in Chapter 5 is applied. The estimated trajectories in the three estimation stages (RawSK, KF-SK, PostSK) and the ground truth trajectory are plotted in Figure 6.38.

After the joint state estimator with the outlier component rejection mechanism, most of the outliers in the KF-SK are rejected successfully. Then, after the motion smoother, the residual outliers are wiped out, and the trajectory becomes smoother in PostSK. In addition, the PostSK is also close to the ground truth after the proposed 3D skeleton modification method.

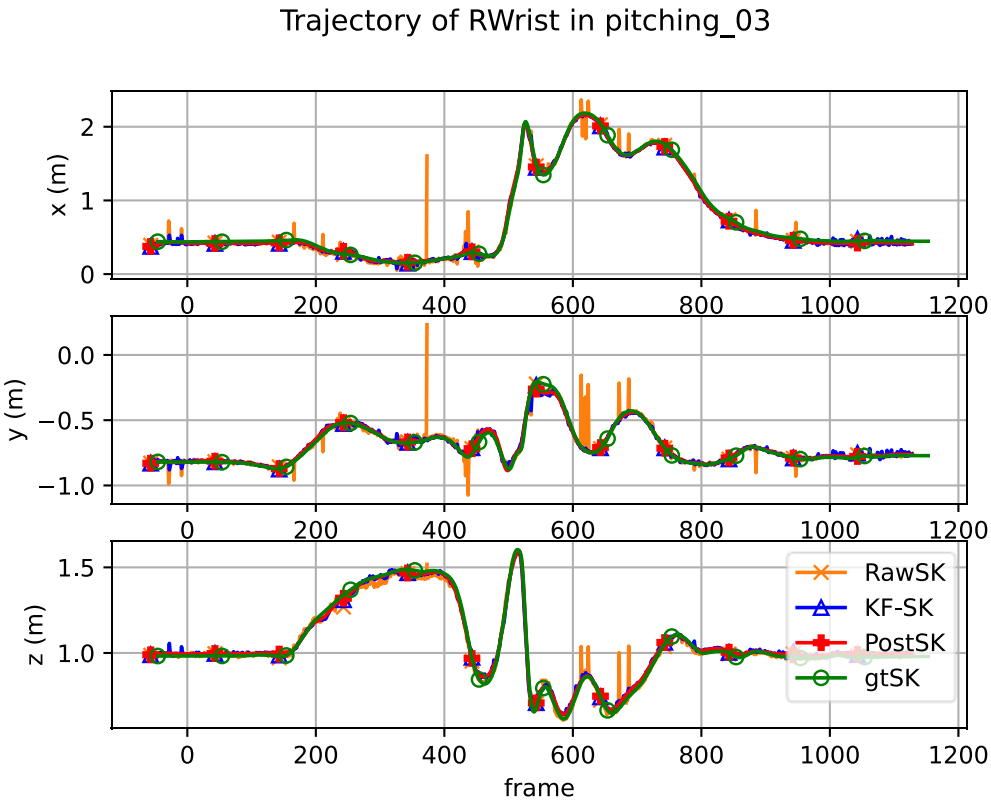
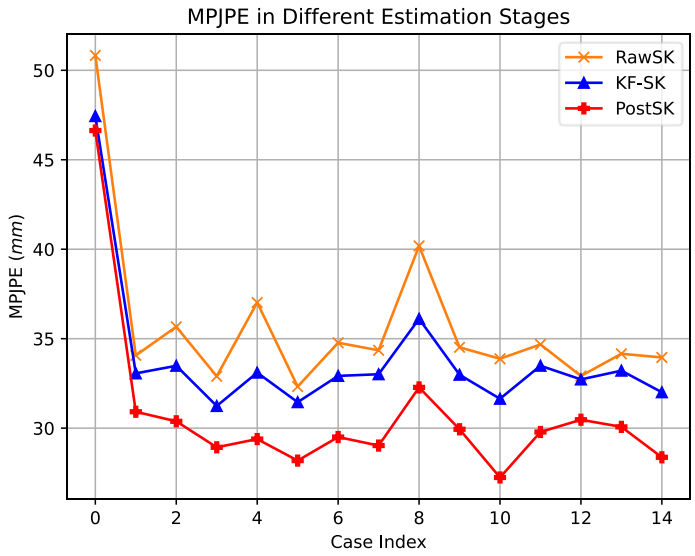


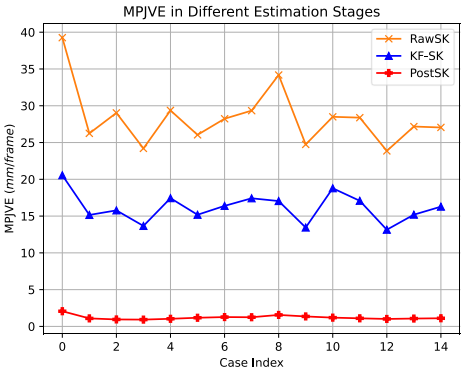
Figure 6.38: Example of keypoint trajectory in different estimation stages

doi:10.6342/NTU202202468

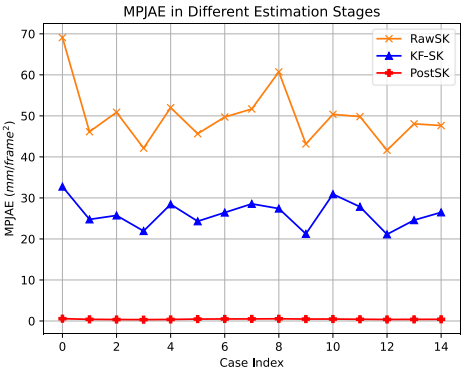
The numerical errors are illustrated in Figure 6.39. No matter in MPJPE, MPJVE, or MPJAE, the numerical errors decrease gradually for all the cases in the experiment. However, the final performances are still affected by the RawSKs. As shown in Figure 6.39 (a), the MPJPE is still high in Hitting_01 after the modification. Also, the MPJPEs for KF-SK and PostSK look highly relative to the MPJPE for RawSKs.



(a)



(b)



(c)

Figure 6.39: Estimation error in different estimation stages
(a) MPJPE (b) MPJVE (c) MPJAE

doi:10.6342/NTU202202468

The average errors for every case in different estimation stages are listed in Table 6.11. For the MPJPE, the errors decrease gradually in the three estimation stages. For MPJVE and MPJAE, the errors drop dramatically in proportion between the KF-SK and the PostSK, which shows that the motion smoother works.

Table 6.11

Average Estimation Performance in the Experiment

Skeleton $\mathcal{S}$	$E_{MPJPE}(\mathcal{S})$ Unit: (mm)	$E_{MPJVE}(\mathcal{S})$ Unit: (mm/frame)	$E_{MPJAE}(\mathcal{S})$ Unit:(mm/frame^2)
RawSK	35.75	28.37	49.91
KF-SK	33.86	16.17	26.17
PostSK	30.74	1.21	0.44

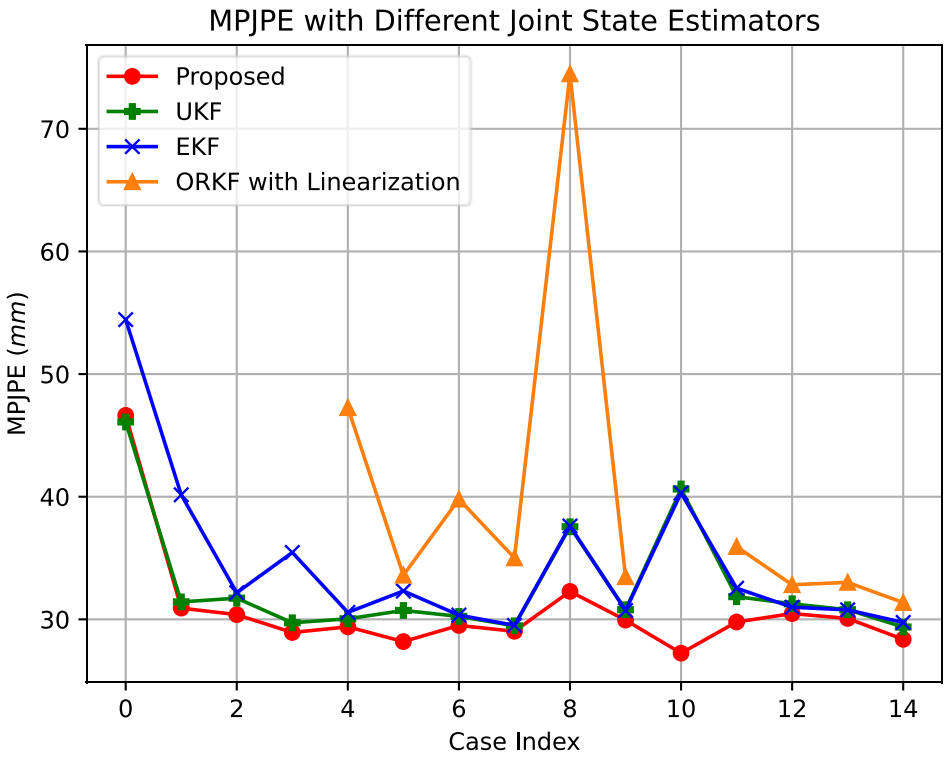
doi:10.6342/NTU202202468

6.6.3 Effectiveness of OCR Joint State Estimator

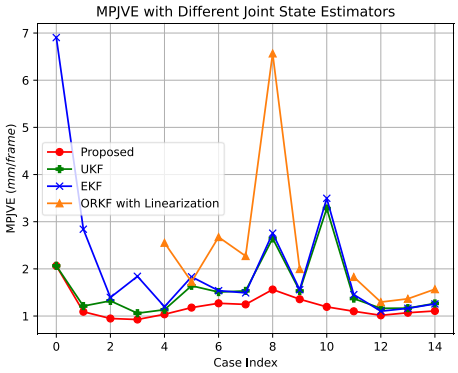
Same as the simulation, the proposed joint state estimator is going to be compared with the other joint state estimators: extended Kalman filter (EKF), unscented Kalman filter (UKF), and the outlier-robust Kalman filter (ORKF) proposed in **Algorithm 1** in [56: Agamennoni et al. 2011]. The concept of the ORKF is that it assumes the covariance of measurement noise is time-varying. Then, the time-varying measurement noise covariance $\mathbf{\Gamma}_t$ is estimated with an iterative method at each frame. With the mechanism, the outliers can be assigned with higher $\mathbf{\Gamma}_t$. Therefore, the influence of outliers would be reduced. However, since the ORKF is designed for linear systems, the human kinematic model doesn't fit it. To fulfill the implementation, the linearization method same as EKF is applied to ORKF to avoid this problem.

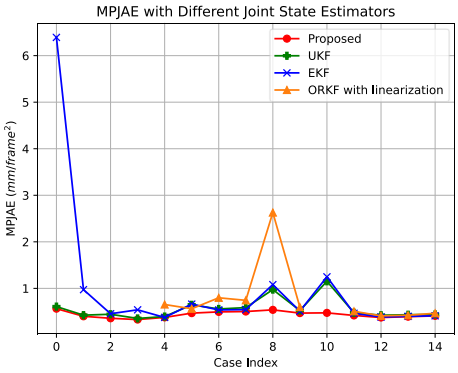
In Figure 6.40, the estimated errors of the proposed method with different joint state estimators are shown. The proposed OCR joint state estimator outperforms the other three joint state estimators in most cases except Hitting_01. In Hitting_01, lots of the keypoint positions are incorrect and far from the ground truth as mentioned in Section 6.6.1. Because of this reason, the outlier component detection mechanism may not work properly and cause the results. For ORKF, there are some missed data because $\mathbf{\Gamma}_t$ didn't converge during the iterative $\mathbf{\Gamma}_t$ estimation. Therefore, the ORKF can't work for the cases.

doi:10.6342/NTU202202468



(a)





(b) (c)

Figure 6.40: The performance different joint state estimator in experiment
(a) MPJPE (b) MPJVE (c) MPJAE

doi:10.6342/NTU202202468

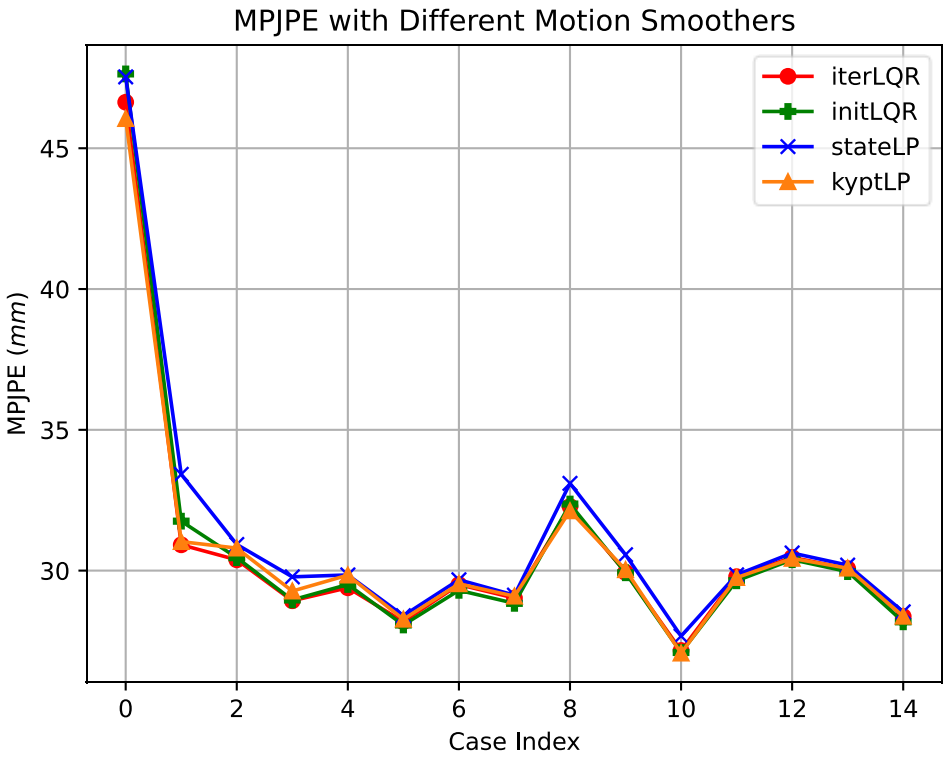
6.6.4 Effectiveness of Iterative LQR Motion Smoother

The comparison of different motion smoothers will be discussed in this section. The compared motion smoothers are the same as those in the simulation: the proposed iterative LQR tracking method (iterLQR), the initial smoothing method of iterLQR (initLQR), the smoothing method with low pass filter on joint state (stateLP), and the smoothing method with low pass filter on keypoint positions (kyptLP).

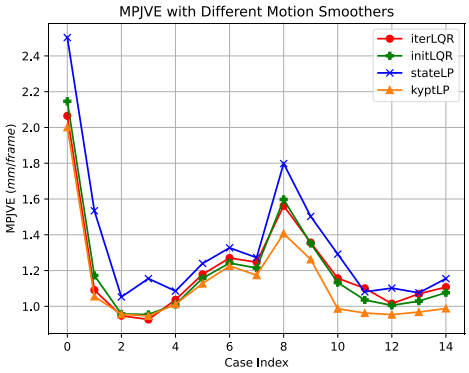
The performances are shown in Figure 6.41. For MPJPE, the differences among the four motion smoothers are not obvious. For MPJVE, the performance of kyptLP is slightly better than the others. The stateLP is the worst. The iterLQR and initLQR perform similarly. Finally, for MPJAE, the proposed iterLQR has the worst performance among the motion smoothers.

But, in fact, the differences between these motion smoothers are subtle. It's more important to ensure whether the proposed iterLQR works for the designed purpose, such as increasing the endpoint position accuracy and maintaining the segment lengths. Therefore, the position errors for the endpoints in the human skeleton are illustrated in Figure 6.42.

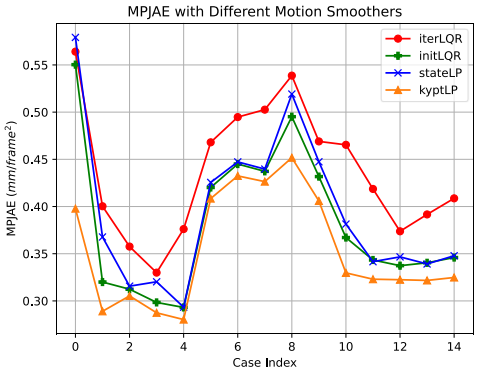
doi:10.6342/NTU202202468



(a)



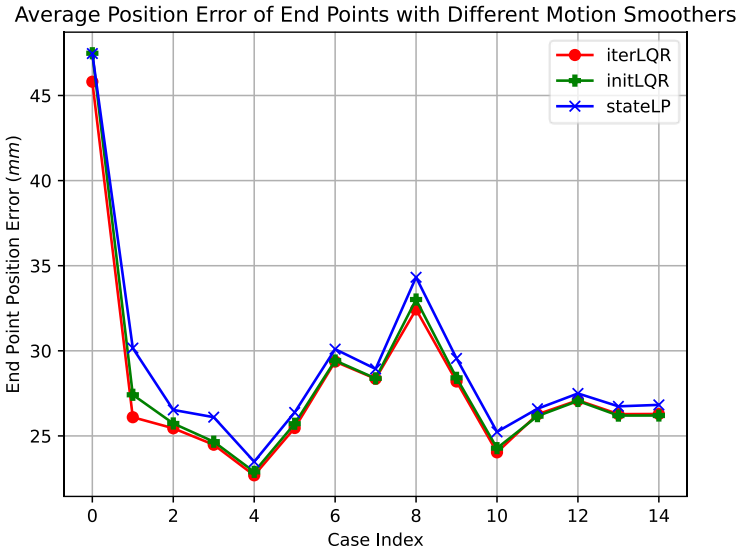
(b)



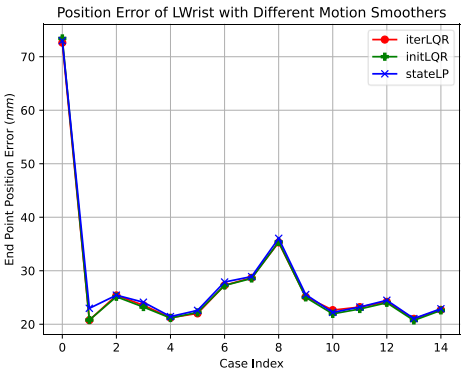
(c)

Figure 6.41: The performance different motion smoothers in experiment
(a) MPJPE (b) MPJVE (c) MPJAE

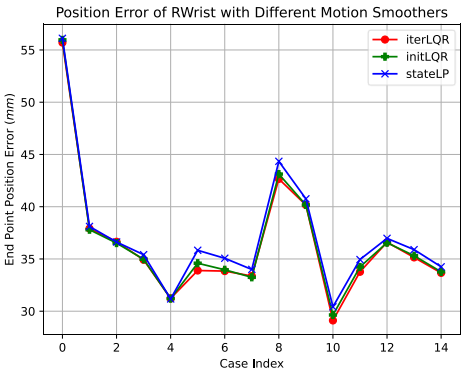
doi:10.6342/NTU202202468



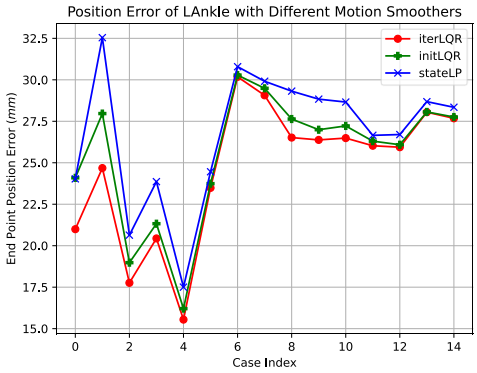
(a)



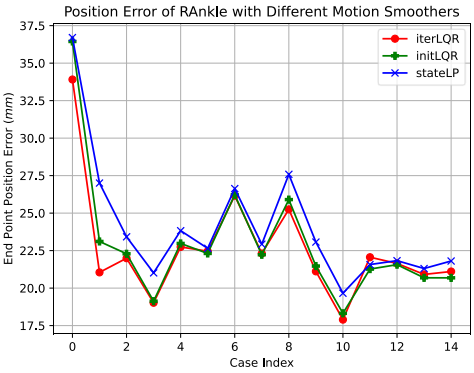
(a)



(b)



(c)



(d)

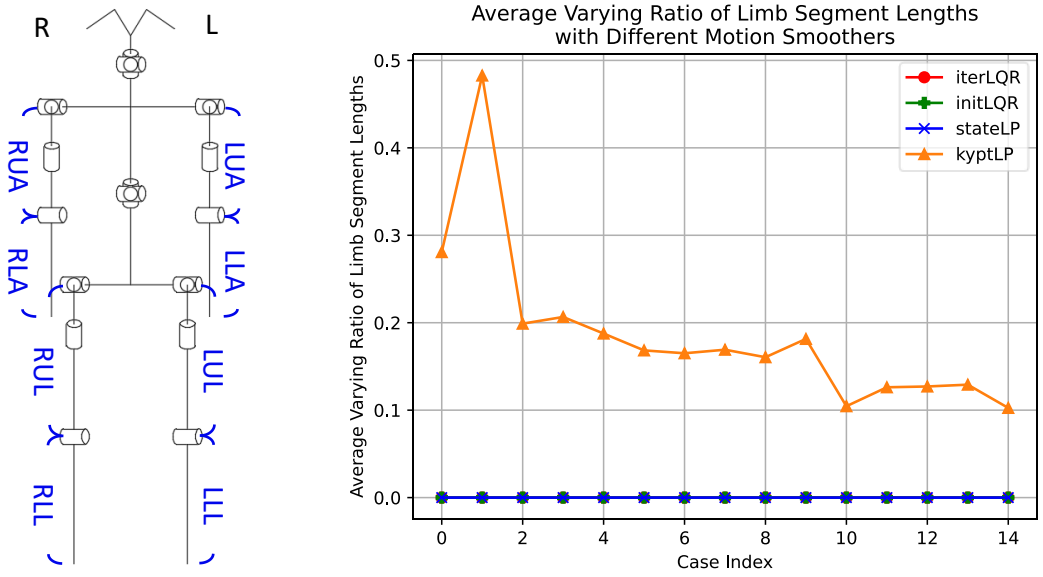
Figure 6.42: The end point position errors with different state-filtering based motion smoothers

(a) Average position error of four end points (b) Position error of LWrist (c) Position error of RWrist (d) Position error of LAnkle (e) Position error of RAnkle

doi:10.6342/NTU202202468

As Figure 6.42 shown, the proposed iterLQR outperforms the other state-filtering-based methods in most cases, especially in LAnkle which usually is the fixed anchor during the high-speed motion in the experiment.

In Figure 6.43, we take the eight segments on the four limbs, as Figure 6.43 (a) shows, to calculate the average varying ratio of the segment lengths in the experiment. As shown in Figure 6.43 (b), the varying ratios of kyptLP can be high over 0.4 and at least 0.1 in the experiment. At the same time, the other motion smoothers maintain 0 varying ratios for the segment lengths since they consider the human skeleton kinematic model.



(a) (b)

Figure 6.43: Varying Ratios with different motion smoothers in experiment
(a) The segment lengths used to take the average (b) The average varying ratios with different motion smoothers in the experiment

doi:10.6342/NTU202202468

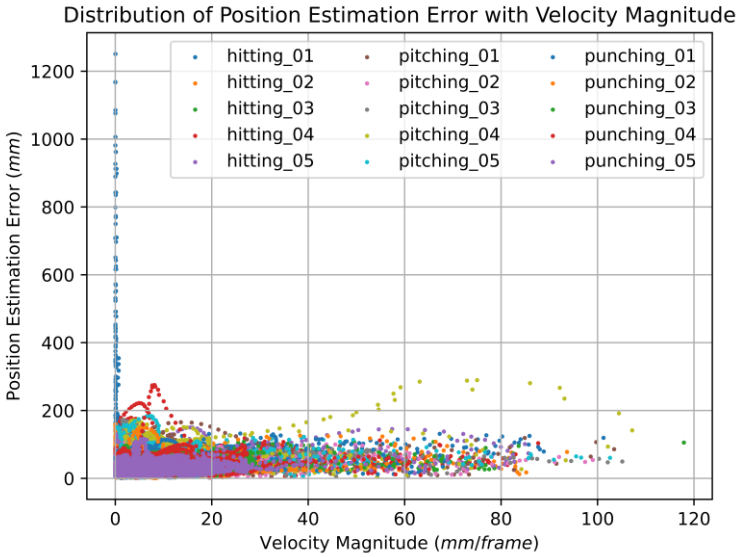
6.6.5 Influence of the Motion Speed

In this section, the influence of the motion speed will be discussed. However, unlike the simulation, we can't adjust the motion speed of the target with one variable directly. Instead, we will analyze the performance with the instantaneous velocity and acceleration magnitudes for every keypoint in each frame.

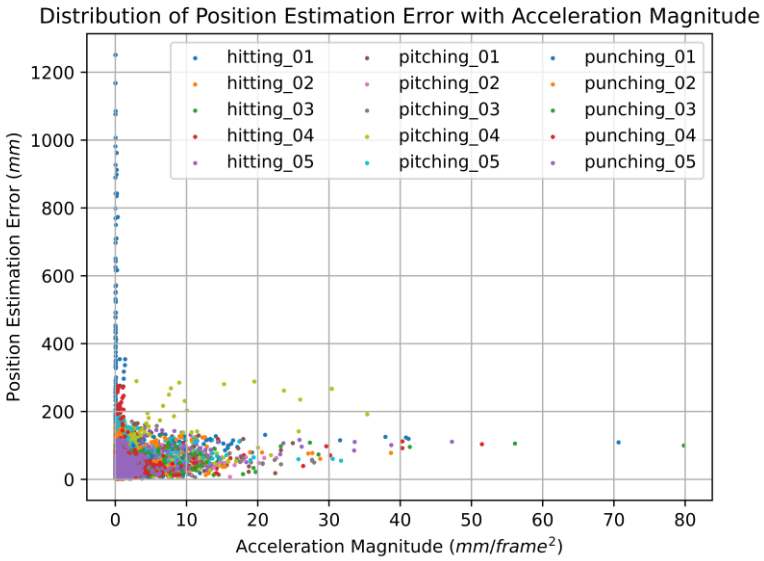
The scatter plots of the position, velocity, and acceleration estimation errors with respect to the keypoint instantaneous velocity and acceleration magnitudes are illustrated in Figure 6.44, Figure 6.45, and Figure 6.46. We can observe that there seem to be some relationships, but some other factors, such as poor skeleton initialization, have significant impacts on a small number of keypoints.

To get rid of these impacts, we divide the keypoints into ten intervals with velocity magnitudes and acceleration magnitudes, as shown in Figure 6.47. In the experiment, most keypoints are concentrated around the low-velocity intervals and the low acceleration intervals. The numbers of the keypoints in the intervals are annotated in Figure 6.47. There are 17973 frames in the 15 cases totally. Since there are 12 body keypoints to be compared, the total number of the keypoints is 215676. The vast majority of the data came from the first intervals of velocity and acceleration magnitude.

doi:10.6342/NTU202202468



(a)

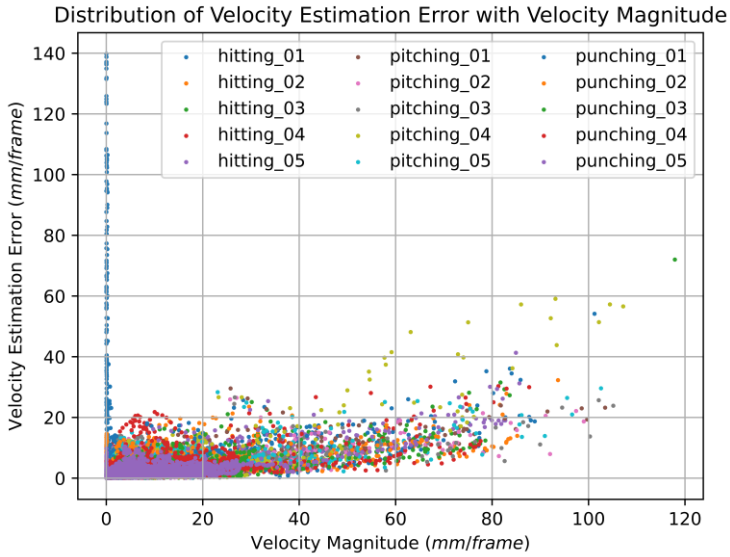


(b)

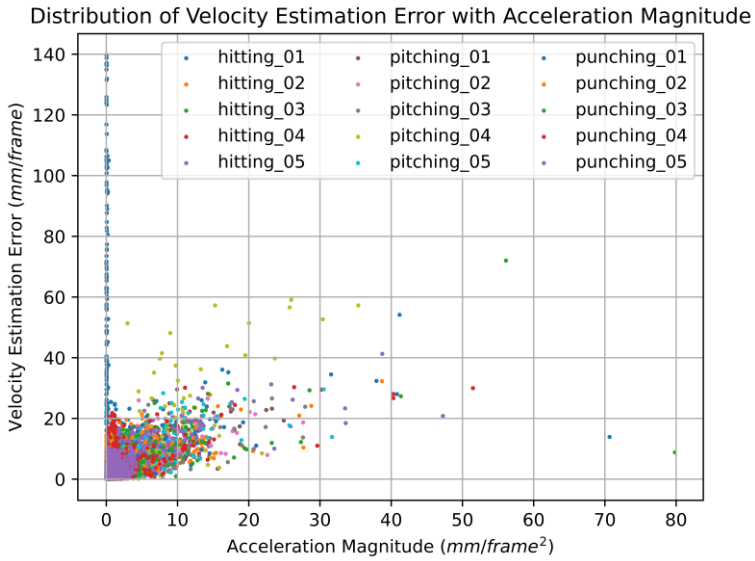
Figure 6.44: Position Estimation Error Distribution with Velocity/Acceleration Magnitude of Keypoints

(a) Position Estimation Error with Velocity Magnitude (b) Position Estimation Error with Acceleration Magnitude

doi:10.6342/NTU202202468



(a)

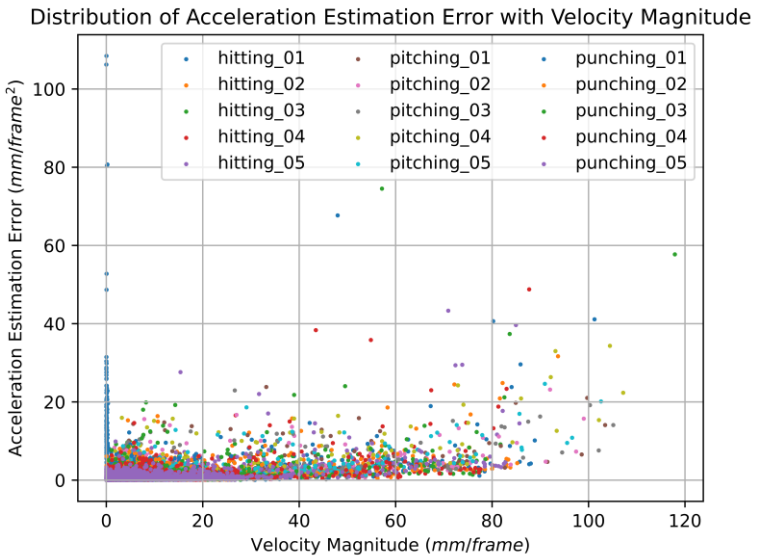


(b)

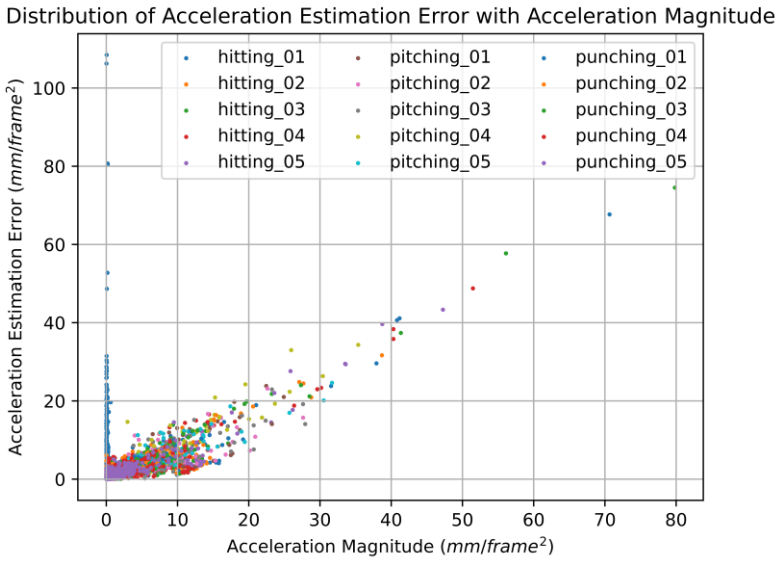
Figure 6.45: Velocity Estimation Error Distribution with Velocity/Acceleration Magnitude of Keypoints

(a) Velocity Estimation Error with Velocity Magnitude (b) Velocity Estimation Error with Acceleration Magnitude

doi:10.6342/NTU202202468



(a)

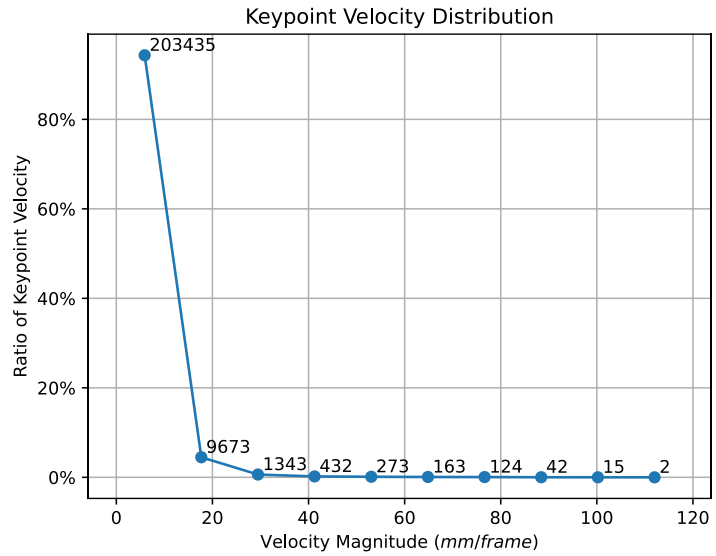


(b)

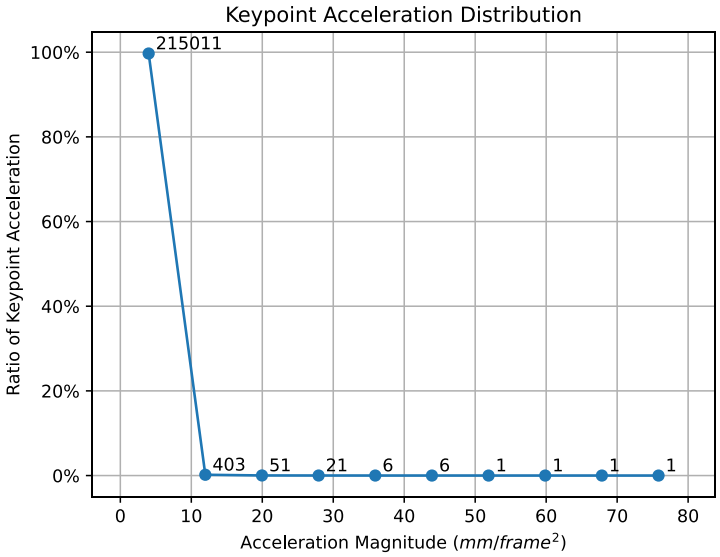
Figure 6.46: Acceleration Estimation Error Distribution with Velocity/Acceleration Magnitude of Keypoints

(a) Acceleration Estimation Error with Velocity Magnitude (b) Acceleration Estimation Error with Acceleration Magnitude

doi:10.6342/NTU202202468



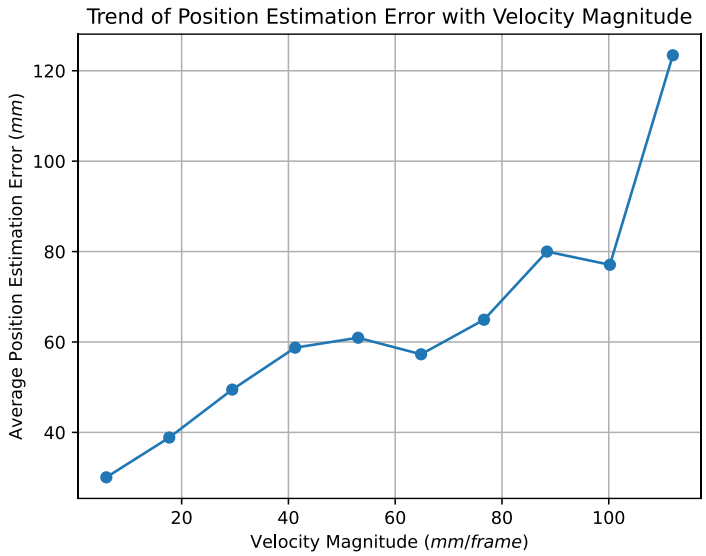
(a)



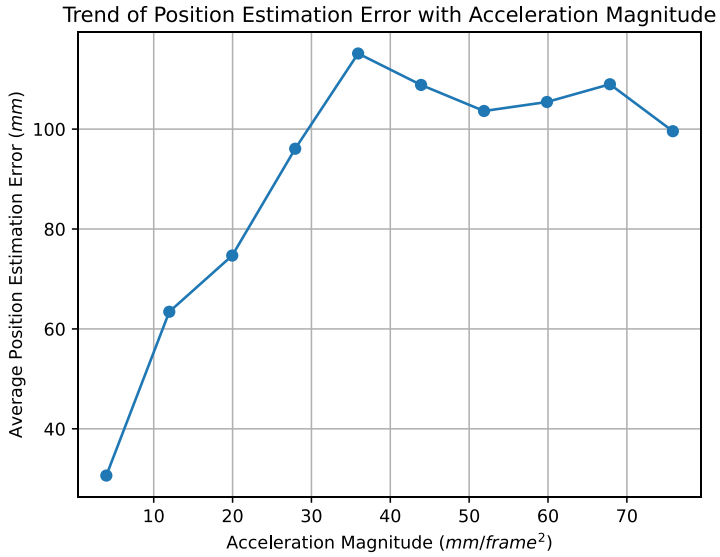
(b)

Figure 6.47: Dynamic distribution of the keypoints
(a) Velocity distribution (b) Acceleration distribution

doi:10.6342/NTU202202468



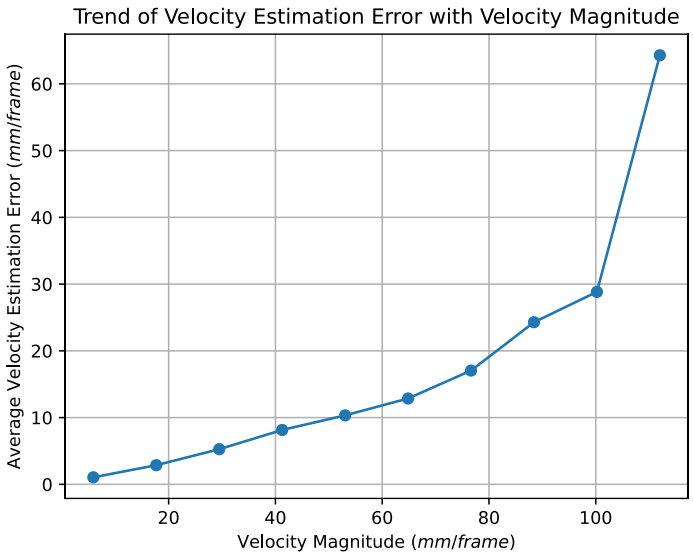
(a)



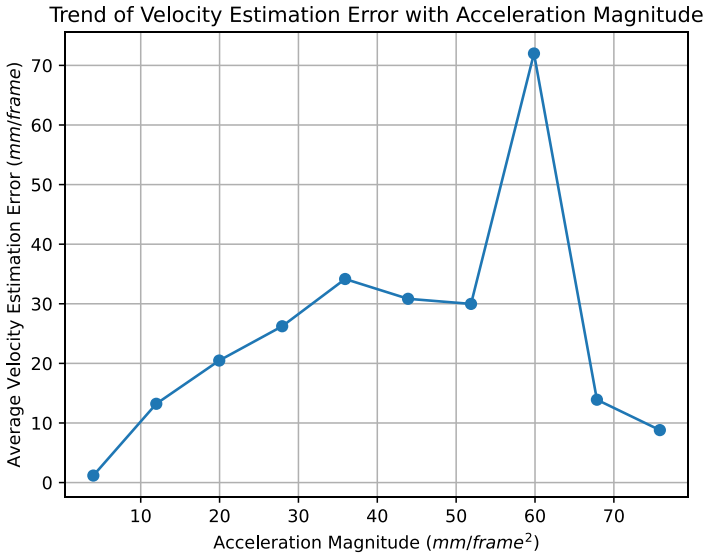
(b)

Figure 6.48: Position Estimation Error with Velocity/Acceleration Magnitude of Keypoints
(a) Position Estimation Error with Velocity Magnitude (b) Position Estimation Error with Acceleration Magnitude

doi:10.6342/NTU202202468



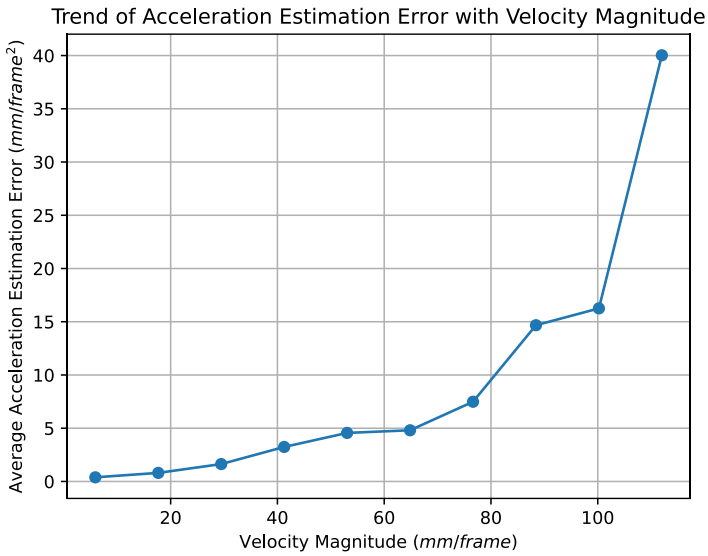
(a)



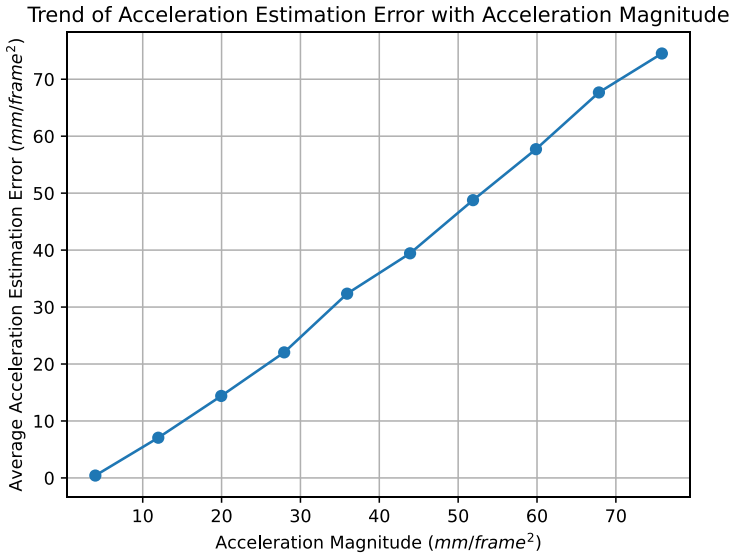
(b)

Figure 6.49: Velocity Estimation Error with Velocity/Acceleration Magnitude of Keypoints
(a) Velocity Estimation Error with Velocity Magnitude (b) Velocity Estimation Error with Acceleration Magnitude

doi:10.6342/NTU202202468



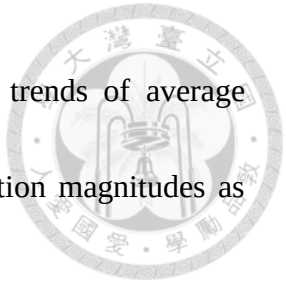
(a)



(b)

Figure 6.50: Acceleration Estimation Error with Velocity/Acceleration Magnitude of Keypoints
(a) Acceleration Estimation Error with Velocity Magnitude (b) Acceleration Estimation Error with Acceleration Magnitude

doi:10.6342/NTU202202468



Taking the average errors for each interval, we can get the trends of average performances with respect to the velocity magnitudes and acceleration magnitudes as shown in Figure 6.48, Figure 6.49, and Figure 6.50.

Except for the acceleration intervals whose values are greater 30 $mm/frame^2$, the estimation errors show obvious trends to increase with the dynamic magnitudes, no matter for velocity or acceleration. It appears the high relationships between the estimation errors and the motion dynamic. The relationships can also correspond to the weakness of the constant velocity model in joint space. The ignored joint accelerations may be unignorable when the motion dynamic increases.

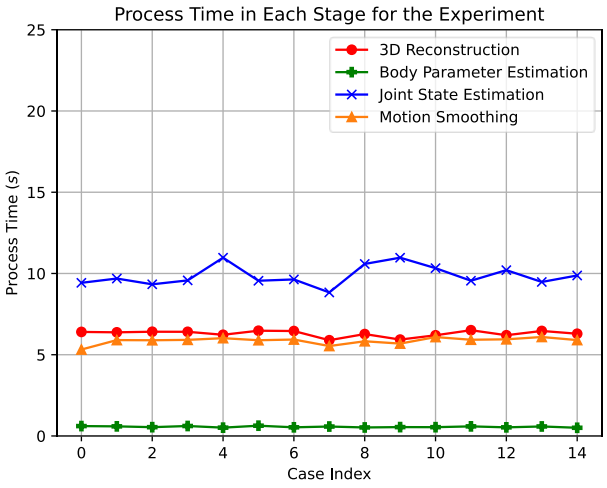
For the acceleration intervals whose values are greater 30 $mm/frame^2$, the numbers of sampled keypoints are too small. Therefore, the results are vulnerable and not informative.

doi:10.6342/NTU202202468

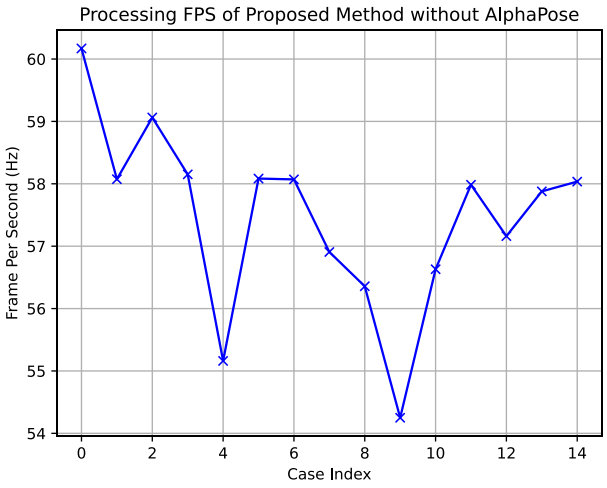
6.6.6 Processing Time for Proposed Method

After the performance analysis, the consumed time is also a consideration for practice. The number of processed frames for a case is from 1186 to 1311 frames in the experiments. The processing times for each stage are illustrated in Figure 6.51 (a). The average processing times are 6.30 seconds for 3D reconstruction, 0.56 seconds for body parameter estimation, 9.88 seconds for joint state estimation, and 5.86 seconds for motion smoothing. The processing FPS without the running time of AlphaPose in the experiment is plotted in Figure 6.51 (b). The average FPS is 57.47 Hz. All the estimated results were processed with Intel Core i9-10900 CPU @ 2.8GHz.

doi:10.6342/NTU202202468



(a)



(b)

Figure 6.51: Processing time for proposed method
(a) processing time for each stage (b) FPS without AlphaPose

doi:10.6342/NTU202202468

6.7 Compare with Deep learning-based methods

As mentioned in Section 2.2.2, the works for 3D human motion estimation in recent years are mainly based on deep learning. To present a more complete comparison, the performance difference between the proposed method and other deep learning approaches will be shown in this section. Before the numerical results, how to evaluate the proposed method with those methods on an approximate benchmark will be explained in Section 6.7.1. Finally, the specific performances for different actions will be listed in Section 6.7.2.

6.7.1 Human3.6M Evaluation Setups

To compare the performance of 3D human motion estimation or 3D human pose estimation for single-person, the most famous dataset is Human3.6M [29: Ionescu et al. 2014]. The most commonly used comparison benchmark is Protocol 1 for multi-view estimation mentioned in [70: Fang et al. 2018], which is to use subjects 1,5,6,7,8 for training and subjects 9 and 11 for evaluation. There are 17 keypoints used to evaluate as shown in Figure 6.52. Naturally, the proposed method can't be implemented on the protocol since the keypoint definition in Human3.6M doesn't fit the skeleton kinematic model illustrated in Figure 5.1. In addition, the positions of the keypoints on the limbs are slightly different from AlphaPose skeleton. For example, $\boldsymbol{p}_0$ and $\boldsymbol{p}_5$ are labeled on heels not ankles.

doi:10.6342/NTU202202468

Figure 6.52: Human skeleton defined in Human3.6M

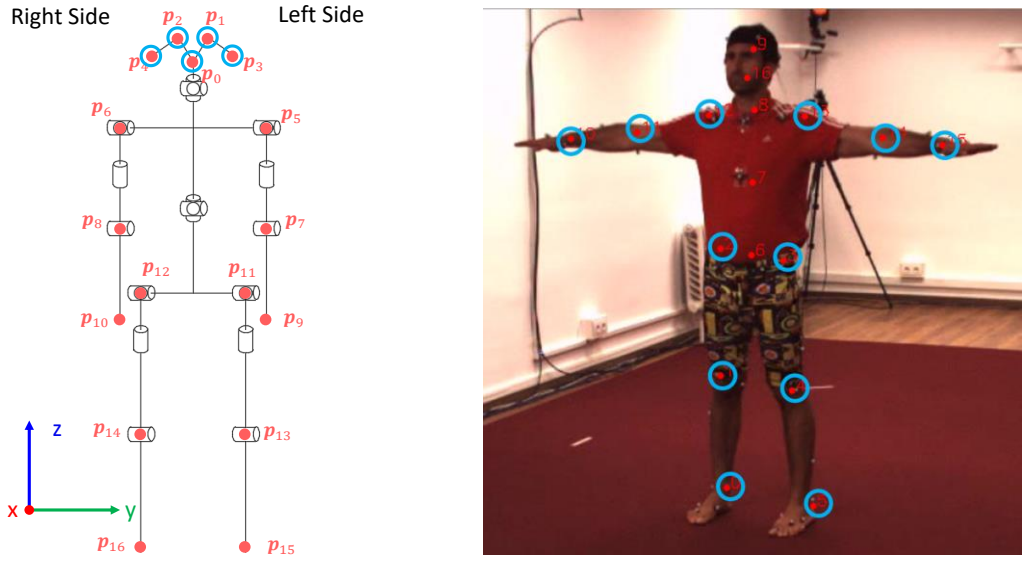
To handle the keypoint defining issue, we use another 3D human pose estimation model, Learnable Triangulation [41: Iskakov et al. 2019], trained on Human3.6M to reconstruct the RawSKs instead of AlphaPose.

Learnable Triangulation, like AlphaPose, is a model that only cares about position errors and didn't consider the temporal information in the motion sequences. Therefore, it's potential to increase the velocity and acceleration performance with the proposed method.

As shown in Figure 6.53, for the implementation, the RawSKs are composed of the keypoints marked with blue circles. Since there is no corresponding keypoint for the same head structure, we use the reconstructed head keypoints from AlphaPose in Figure 6.53 (a) to build the head part of the RawSKs. Then, for the body part, we use the keypoints

doi:10.6342/NTU202202468

on the limbs from Learnable Triangulation to construct the rest parts of RawSKs. The slight differences in keypoint definition are regarded as the differences in body parameters. Eventually, the proposed method can be used on Human3.6M with the same parameter settings in Section 6.5.3.



(a) (b)

Figure 6.53: The keypoint definition for RawSK for Human3.6M
(a) The keypoints of head come from the reconstructed 3D AlphaPose (b) The keypoints of body come from the output of Learnable Triangulation trained on Human3.6M

For the evaluation, since there is no corresponding ground truth for the head keypoints, the evaluated results will only compare the 12 keypoints errors on the four limbs as shown in Figure 6.53 (b), which makes the benchmark slightly different than the benchmarks in other works. Besides, because the evaluated keypoints in RawSKs are all from Learnable Triangulation, the evaluated errors of the RawSK can be regarded as the performance of Learnable Triangulation.

doi:10.6342/NTU202202468

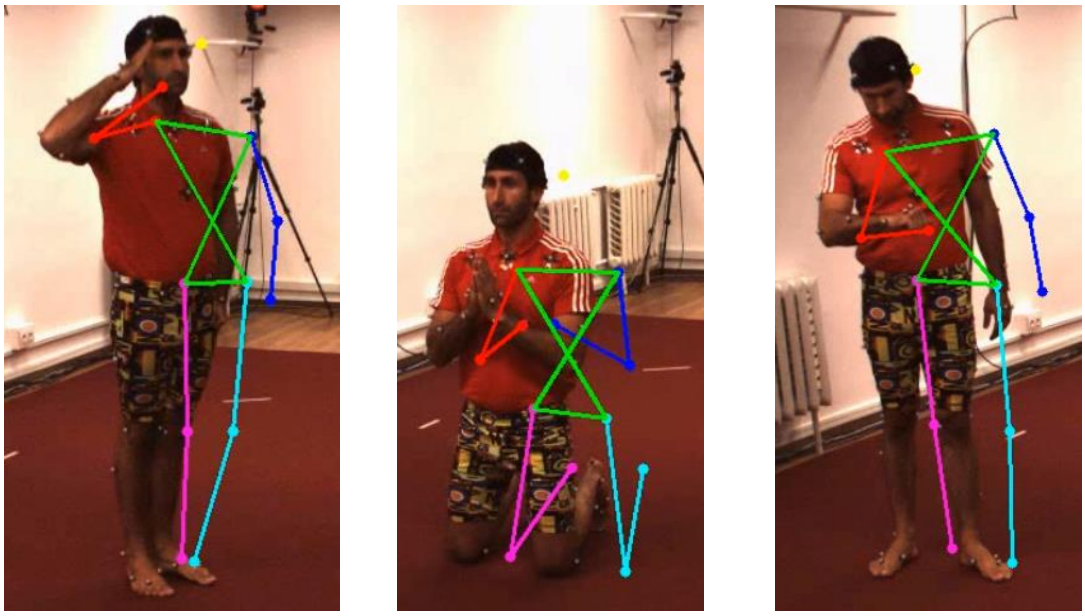
Table 6.12

Tested Cases with Human3.6M

Subject	Action	Case Name
S9/S11	Directions	Directions
		Directions_1
	Discussion	Discussion_1
		Discussion_2
	Eating	Eating
		Eating_1
	Greeting	Greeting
		Greeting_1
	Phoning	Phoning
		Phoning_1
	Posing	Posing
		Posing_1
	Purchases	Purchases
		Purchases_1
	Sitting	Sitting
		Sitting_1
	SittingDown	SittingDown
		SittingDown_1
	Smoking	Smoking
		Smoking_1
	Photo	Photo
		Photo_1
	Waiting	Waiting
		Waiting_1
	Walking	Walking
		Walking_1
	WalkDog	WalkDog
		WalkDog_1
	WalkTogether	WalkTogether
		WalkTogether_1

doi:10.6342/NTU202202468

In Protocol 1, the tested cases are listed in Table 6.12. However, as the camera parameters in S9_Greeting_1, S9_SittingDown_1, and S9_Waiting are incorrect, the three cases will be eliminated for evaluation. The projected ground truth skeletons with the wrong camera parameters in the three cases are shown in Figure 6.54.



(a) (b) (c)

Figure 6.54: Projected ground truth skeleton in the eliminated cases on Human3.6M (a) S9_Greeting_1 (b) S9_SittingDown_1 (c) S9_Waiting

The rest of the cases will be used to evaluate the estimation performance numerically.

doi:10.6342/NTU202202468

6.7.2 Performance Analysis of Motion Estimation

In this section, the numerical errors will be displayed. For position evaluation (MPJPE), the works are divided into two groups: considering velocity errors or not considering velocity errors.

Many works put a lot of effort and focus on MPJPE results during the development because the original evaluation metrics for Human3.6M only consider the position performance. However, in recent years, more and more papers claimed that position is not the only thing determining the estimation performance by humans. Therefore, the MPJVE had come out to provide a higher-order metric for human motion estimation.

The works considering velocity errors are the papers presenting their MPJPE and MPJVE at the same time and are listed in Table 6.13. For the works focusing on the position performance, their MPJPEs are listed in Table 6.14.

doi:10.6342/NTU202202468

Table 6.13

Comparison for Position Error (MPJPE) in mm

with method considering velocity errors on Human3.6M

Action	RawSK (Learnable Triangulation)	PostSK	[60: Pavllo et al. 2019]	[61: Lin & Lee 2019]	[62: Chen et al. 2022]	[63: Wang et al. 2020]	[64: Li et al. 2022]
Directions	18.5	**29.6**	45.2	42.5	41.4	40.2	40.3
Discussion	19.1	**24.5**	46.7	44.8	43.5	42.5	43.3
Eating	19.8	**25.0**	43.3	42.6	40.1	42.6	40.2
Greeting	18.8	**25.7**	45.6	44.2	42.9	41.1	42.3
Phoning	21.2	**28.7**	48.1	48.5	46.6	46.7	45.6
Posing	18.5	**25.2**	44.6	42.6	41.7	41.4	41.8
Purchases	18.6	**31.1**	44.3	41.4	42.3	42.3	40.5
Sitting	22.0	**27.7**	57.3	56.5	53.9	56.2	55.9
SittingDown	20.6	**28.0**	65.8	64.5	60.2	60.4	60.6
Smoking	21.1	**26.6**	47.1	47.4	45.4	46.3	44.2
Photo	22.1	**27.0**	55.1	57.1	51.9	56.7	52.3
Waiting	18.3	**26.7**	44.0	43.0	41.7	42.2	43.0
Walking	21.5	35.8	32.8	33.0	31.5	31.7	**30.0**
WalkDog	20.0	**31.2**	49.0	48.1	46.0	46.2	44.2
WalkTogether	21.9	35.6	33.9	35.1	32.7	31.0	**30.2**
Mean	20.1	**28.6**	46.8	46.6	44.1	44.5	43.7

As shown in Table 6.13, the proposed method with Learnable Triangulation performs the best position estimation with other velocity considering methods except the action Walking and WalkTogether. One of the reasons for the great performance comes from the fine detection of Learnable Triangulation. But, as with the other velocity considering methods, the combination of the proposed method and Learnable Triangulation harms the position performance because of the trade-off between position and velocity.

doi:10.6342/NTU202202468

Table 6.14

Comparison for Position Error (MPJPE) in mm

with method not considering velocity errors Human3.6M

Action	RawSK (Learnable Triangulation)	PostSK	[65: Bouazizi et al. 2021]	[66: Qiu et al. 2019]	[67: Gordon et al. 2022]	[42: He et al. 2020]	[68: Reddy et al. 2021]
Directions	18.5	29.6	48.2	28.9	22.0	25.7	**17.5**
Discussion	**19.1**	24.5	49.3	32.5	23.6	27.7	19.6
Eating	19.8	25.0	46.5	26.6	24.9	23.7	**17.2**
Greeting	18.8	25.7	48.4	28.1	26.7	24.8	**18.3**
Phoning	21.2	28.7	52.4	28.3	30.6	26.9	**18.2**
Posing	18.5	25.2	46.4	28.0	25.1	24.9	**18.0**
Purchases	18.6	31.1	61.4	36.8	32.9	26.5	**18.0**
Sitting	22.0	27.7	72.3	42.0	29.5	28.8	**20.5**
SittingDown	20.6	28.0	51.0	30.5	32.5	31.7	**20.3**
Smoking	21.1	26.6	59.8	35.6	32.6	28.2	**19.4**
Photo	22.1	27.0	46.5	29.3	35.7	31.4	**17.7**
Waiting	18.3	26.7	46.7	30.0	26.5	26.4	**17.2**
Walking	21.5	35.8	52.1	30.0	26.0	28.3	**18.9**
WalkDog	20.0	31.2	37.5	28.3	34.7	23.6	**19.0**
WalkTogether	21.9	35.6	39.1	30.5	27.7	23.5	**17.8**
Mean	20.1	28.6	50.6	31.2	30.2	26.9	**18.7**

Compared with the works focusing on position performance shown in Table 6.14, the proposed method still has room to improve.

doi:10.6342/NTU202202468

Table 6.15

Velocity Error (MPJVE) in $mm/frame$ on Human3.6M

Action	RawSK (Learnable Triangulation)	PostSK	[60: Pavllo et al. 2019]	[61: Lin & Lee 2019]	[62: Chen et al. 2022]	[63: Wang et al. 2020]	[64: Li et al. 2022]
Directions	2.4	**2.3**	3.0	2.7	2.7	**2.3**	2.4
Discussion	2.6	**2.2**	3.1	2.8	2.8	2.5	2.5
Eating	2.2	1.9	2.2	2.1	2.0	2.0	**1.8**
Greeting	2.8	**2.5**	3.4	3.1	3.1	2.7	2.8
Phoning	2.2	**1.7**	2.3	2.0	2.0	2.0	1.8
Posing	**2.1**	**2.1**	2.7	2.5	2.4	2.2	2.2
Purchases	**2.5**	2.9	3.1	2.9	2.8	**2.5**	**2.5**
Sitting	2.4	**1.2**	2.1	1.8	1.8	1.8	1.5
SittingDown	2.9	**1.4**	2.9	2.6	2.4	2.7	2.0
Smoking	2.1	**1.5**	2.3	2.1	2.0	1.9	1.8
Photo	2.5	**2.1**	2.7	2.5	2.4	2.3	2.2
Waiting	2.0	**1.6**	2.4	2.3	2.1	2.0	1.9
Walking	3.0	4.0	3.1	2.7	2.7	**2.2**	2.5
WalkDog	**3.0**	3.5	3.7	3.7	3.4	3.1	3.2
WalkTogether	2.7	3.0	2.8	3.1	2.4	2.5	**2.1**
Mean	2.5	2.3	2.8	2.7	2.5	2.3	**2.2**

For MPJVEs shown in Table 6.15, the proposed method has the best performance for most of the cases. However, its MPJVE is relatively high for some actions, such as Walking, WalkDog, and WalkTogether. In the three actions, the trajectories of the targets are continuously circling. The keypoints are also doing the accelerated movement, which is detrimental to the proposed method during the actions. As the result, it seems to be the reason for the poor performance in those actions.

doi:10.6342/NTU202202468

Table 6.16

Acceleration Error (MPJAE) in $mm/frame^2$ on Human3.6M

Action	RawSK (Learnable Triangulation)	PostSK	[60: Pavllo et al. 2019]	[69: Gupta 2020]
Directions	3.3	**0.8**	2.3	2.1
Discussion	3.6	**0.9**	2.6	2.4
Eating	3.1	**0.7**	1.8	1.7
Greeting	3.9	**0.9**	2.7	2.4
Phoning	3.1	**0.7**	2.0	1.9
Posing	2.8	**0.8**	2.1	1.9
Purchases	3.5	**1.2**	2.5	2.3
Sitting	3.6	**0.5**	2.1	1.9
SittingDown	4.3	**0.8**	2.1	2.5
Smoking	3.0	**0.6**	2.3	1.9
Photo	3.6	**0.9**	2.3	2.0
Waiting	2.8	**0.6**	2.1	1.9
Walking	4.0	**1.5**	2.8	2.4
WalkDog	4.1	**1.4**	2.1	2.6
WalkTogether	3.7	**1.0**	2.6	2.2
Mean	3.5	**0.9**	2.4	2.1

Last, the acceleration errors for different methods are listed in Table 6.16. Compared to MPJVE, there are fewer papers that discussed acceleration errors (MPJAE). Among the four estimation results, the proposed method outperforms all the others in every action. It also demonstrates the advantage of the motion smoother.

doi:10.6342/NTU202202468

Chapter 7
Conclusions and Future Works

7.1 Conclusions

People have been researching how to capture 3D human motion accurately for the past two decades. With the advancement of sensing technology, more and more high-accuracy motion capture equipment, such as VICON, has been invented. However, the high accurate equipment often encounters high costs and few applicable environments. In recent years, though many vision-based 3D human motion estimation methods have been developed with deep learning techniques, they still can't get rid of the demand for the 3D labeled data which are hard to access in outdoor environments.

In this thesis, a portable multi-view motion capture system is proposed to overcome the environment constraints with the light capture devices and the simple calibration method. The 3D keypoint detection is achieved without 3D labeled data while using the 2D keypoint detection by AlphaPose and the 3D reconstruction method. Since the loss of 3D labeled data, the reconstructed results are usually worse than the pure deep learning-based methods. Therefore, a 3D skeleton modification method is proposed as a solution to this problem. By considering the skeleton kinematics and the proposed outlier component rejecting mechanism, the position estimation has significant improvement.

doi:10.6342/NTU202202468

Besides, the iterative LQR motion smoother is also presented in this thesis to further enhance the velocity and acceleration performance.

With the proposed method, the performance has significant improvement in both simulation and experiment. Furthermore, to compare with the pure deep learning-based works in recent years, Learnable Triangulation is regarded as the detector for the proposed method on Human3.6M. The results also show the outstanding performance among the dynamic considering methods in the low-speed actions.

7.2 Future Works

Although the proposed work demonstrates great performance for the single-person single-track 3D human motion estimation, there are still some opportunities to improve the current method.

For increasing the performance, the information of the subject IDs may be useful to estimate the more accurate body parameters when there are many tested cases for the same persons. Furthermore, while the tested motions are highly repeated and have few modes, such as pitching, the dynamic probability model may fit to estimate the current motion with the information in previous cases.

For the application, online implementation is a potential topic. Since the proposed method is offline due to the iterative LQR motion smoother and body parameter

doi:10.6342/NTU202202468

estimation, the user may not get the most immediate information. The proposed method has the potential to be turned into an online method using the model-predictive approach.

Last, as the proposed method is designed for single-person scenarios, the estimation result may be poor when the target is not isolated from other persons. However, with high discernment of human-ID models, the proposed method can work even in a crowded area. Moreover, since the proposed method has the ability to handle the missing data problem coming from occlusion, multi-person 3D human motion estimation is also a feasible topic with the proposed method.

doi:10.6342/NTU202202468

References

[1: van der Kruk & Reijne 2018]

Eline van der Kruk and Marco M. Reijne, "Accuracy of human motion capture systems for sport applications; state-of-the-art review," European Journal of Sport Science, Vol. 18, No. 6, pp. 806–819, Jul. 2018.

[2: Zheng et al. 2022]

Ce Zheng, Wenhan Wu, Chen Chen, Taojiannan Yang, Sijie Zhu, Nasser Kehtarnavaz, and Mubarak Shah, "Deep Learning-Based Human Pose Estimation: A Survey," arXiv:2012.13392 [cs], Jan. 2022.

[3: Stelzer et al. 2004]

A. Stelzer, K. Pourvoyeur, and A. Fischer, "Concept and application of LPM - a novel 3-D local position measurement system," IEEE Transactions on Microwave Theory and Techniques, Vol. 52, No. 12, pp. 2664–2669, Dec. 2004.

[4: Perrat et al. 2015]

Perrat, Martin J Smith, Barry S Mason, James M Rhodes, and Vicky L Goosey-Tolfrey, "Quality assessment of an Ultra-Wide Band positioning system for indoor wheelchair court sports," Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology, Vol. 229, No. 2, pp. 81–91, Jun. 2015.

[5: Schepers & Veltink 2010]

Martin Schepers and Peter Veltink, "Stochastic magnetic measurement model for relative position and orientation estimation," Measurement Science and Technology, Vol. 21, p. 065801, Apr. 2010.

[6: Sathyan et al. 2012]

Thuraiappah Sathyan, Richard Shuttleworth, Mark Hedley, and Keith Davids, "Validity and reliability of a radio positioning system for tracking athletes in indoor and outdoor team sports," Behavior Research Methods, Vol. 44, No. 4, pp. 1108–1114, Dec. 2012.

[7: Spörri et al. 2016]

Jörg Spörri, Christian Schiefermüller, and Erich Müller, "Collecting Kinematic Data on a Ski Track with Optoelectronic Stereophotogrammetry: A Methodological Study Assessing the Feasibility of Bringing the Biomechanics Lab to the Field," PLOS ONE, Vol. 11, No. 8, p. e0161757, Aug. 2016.

doi:10.6342/NTU202202468

[8: Begon et al. 2009]
Mickael Begon, Floren Colloud, Vincent Fohanno, Pascal Bahuaud, and Tony Monnet, "Computation of the 3D kinematics in a global frame over a 40 m-long pathway using a rolling motion analysis system," Journal of Biomechanics, Vol. 42, pp. 2649–53, Sep. 2009.

[9: Joukov et al. 2017]
Vladimir Joukov, Josip Ćesić, Kevin Westermann, Ivan Marković, Dana Kulić, and Ivan Petrović, "Human motion estimation on Lie groups using IMU measurements," in Proceedings of 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1965–1972, Sep. 2017.

[10: Chen et al. 2020]
Yawen Chen, Chenglong Fu, Winnie Suk Wai Leung, and Ling Shi, "Drift-Free and Self-Aligned IMU-Based Human Gait Tracking System With Augmented Precision and Robustness," IEEE Robotics and Automation Letters, Vol. 5, No. 3, pp. 4671–4678, Jul. 2020.

[11: Sarafianos et al. 2016]
Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A. Kakadiaris, "3D Human pose estimation: A review of the literature and analysis of covariates," Computer Vision and Image Understanding, Vol. 152, pp. 1–20, Nov. 2016.

[12: Wang et al. 2021]
Jinbao Wang, Shujie Tan, Xiantong Zhen, Shuo Xu, Feng Zheng, Zhenyu He, and Ling Shao, "Deep 3D human pose estimation: A review," Computer Vision and Image Understanding, Vol. 210, p. 103225, Sep. 2021.

[13: Zhang 2012]
Zhengyou Zhang, "Microsoft Kinect Sensor and Its Effect," IEEE MultiMedia, Vol. 19, No. 2, pp. 4–10, Feb. 2012.

[14: Kim et al. 2019]
Wonhui Kim, Manikandasriram Srinivasan Ramanagopal, Charles Barto, Ming-Yuan Yu, Karl Rosaen, Nick Goumas, Ram Vasudevan, and Matthew Johnson-Roberson, "PedX: Benchmark Dataset for Metric 3-D Pose Estimation of Pedestrians in Complex Urban Intersections," IEEE Robotics and Automation Letters, Vol. 4, No. 2, pp. 1940–1947, Apr. 2019.

[15: Du et al. 2019]
Xiaoxiao Du, Ram Vasudevan, and Matthew Johnson-Roberson, "Bio-LSTM: A Biomechanically Inspired Recurrent Neural Network for 3-D Pedestrian Pose and Gait Prediction," IEEE Robotics and Automation Letters, Vol. 4, No. 2, pp. 1501–1508, Apr. 2019.

doi:10.6342/NTU202202468

[16: Hwang et al. 2017]
Jihye Hwang, Sungheon Park, and Nojun Kwak, "Athlete Pose Estimation by a Global-Local Network," in Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 114–121, Jul. 2017.

[17: Rematas et al. 2018]
Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, and Steve Seitz, "Soccer on Your Tabletop," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4738–4747, 2018.

[18: Daubney et al. 2012]
Ben Daubney, David Gibson, and Neill Campbell, "Estimating pose of articulated objects using low-level motion," Computer Vision and Image Understanding, Vol. 116, No. 3, pp. 330–346, Mar. 2012.

[19: Ning et al. 2008]
Wei Ning, Cai Hong-ming, and Jiang Li-hong, "A dynamical adjustment partitioning algorithm for distributed virtual environment systems," in Proceedings of Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry, New York, NY, USA, pp. 1–5, Dec. 8, 2008.

[20: Belagiannis et al. 2014]
Vasileios Belagiannis, Sikandar Amin, Mykhaylo Andriluka, Bernt Schiele, Nassir Navab, and Slobodan Ilic, "3D Pictorial Structures for Multiple Human Pose Estimation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1669–1676, 2014.

[21: Burenius et al. 2013]
Magnus Burenius, Josephine Sullivan, and Stefan Carlsson, "3D Pictorial Structures for Multiple View Articulated Pose Estimation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3618–3625, 2013.

[22: Zuffi et al. 2012]
Silvia Zuffi, Oren Freifeld, and Michael J. Black, "From Pictorial Structures to deformable structures," in Proceedings of 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3546–3553, Jun. 2012.

[23: Huang & Yang 2010]
Jia-Bin Huang and Ming-Hsuan Yang, "Estimating Human Pose from Occluded Images," in Proceedings of Computer Vision – ACCV 2009, Berlin, Heidelberg, pp. 48–60, 2010.

doi:10.6342/NTU202202468

[24: Sedai et al. 2010]

Suman Sedai, Mohammed Bennamoun, and Du Huynh, "Localized fusion of Shape and Appearance features for 3D Human Pose Estimation," in Proceedings of Procedings of the British Machine Vision Conference 2010, Aberystwyth, p. 51.1-51.10, 2010.

[25: Grauman et al. 2003]

Grauman, Shakhnarovich, and Darrell, "Inferring 3D structure with a statistical image-based shape model," in Proceedings of Proceedings Ninth IEEE International Conference on Computer Vision, pp. 641–647 vol.1, Oct. 2003.

[26: Bergh et al. 2009]

Van den Bergh, M., Koller-Meier, E., Kehl, R., Van Gool, L., "Real-Time 3D Body Pose Estimation.," in Multi-Camera Networks: Principles and Applications, pp. 335-361, 2009.

[27: Rosales & Sclaroff 2006]

RÓMer Rosales and Stan Sclaroff, "Combining Generative and Discriminative Models in a Framework for Articulated Pose Estimation," International Journal of Computer Vision, Vol. 67, No. 3, pp. 251–276, May 2006.

[28: Sedai et al. 2013]

S. Sedai, M. Bennamoun, and D.Q. Huynh, "Discriminative fusion of shape and appearance features for human pose estimation," Pattern Recognition, Vol. 46, No. 12, pp. 3223–3237, Dec. 2013.

[29: Ionescu et al. 2014]

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu, "Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 36, No. 7, pp. 1325–1339, Jul. 2014.

[30: Ionescu et al. 2011]

Catalin Ionescu, Fuxin Li, and Cristian Sminchisescu, "Latent structured models for human pose estimation," in Proceedings of 2011 International Conference on Computer Vision, pp. 2220–2227, Nov. 2011.

[31: Joo et al. 2018]

Hanbyul Joo, Tomas Simon, and Yaser Sheikh, "Total Capture: A 3D Deformation Model for Tracking Faces, Hands, and Bodies," arXiv:1801.01615 [cs], Jan. 2018.

[32: Loper et al. 2015]

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black, "SMPL: a skinned multi-person linear model," ACM Transactions on Graphics, Vol. 34, No. 6, p. 248:1-248:16, Oct. 2015.

doi:10.6342/NTU202202468

[33: Güler et al. 2018]

Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos, "DensePose: Dense Human Pose Estimation in the Wild," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7297–7306, 2018.

[34: Luvizon et al. 2018]

Diogo C. Luvizon, David Picard, and Hedi Tabia, "2D/3D Pose Estimation and Action Recognition Using Multitask Deep Learning," pp. 5137–5146, 2018.

[35: Pavlakos et al. 2017]

Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis, "Coarse-To-Fine Volumetric Prediction for Single-Image 3D Human Pose," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7025–7034, 2017.

[36: Nie et al. 2017]

Bruce Xiaohan Nie, Ping Wei, and Song-Chun Zhu, "Monocular 3D Human Pose Estimation by Predicting Depth on Joints," Proceedings of the IEEE International Conference on Computer Vision, pp. 3447–3455, 2017.

[37: Bogo et al. 2016]

Bogo F., Kanazawa A., Lassner C., Gehler P., Romero J., Black M.J, "Keep It SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image," 2016 European Conference on Computer Vision, Springer, pp. 561–578., 2016.

[38: Kanazawa et al. 2018]

Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik, "End-to-End Recovery of Human Shape and Pose," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7122–7131, 2018.

[39: Pavlakos et al. 2017]

Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis, "Harvesting Multiple Views for Marker-Less 3D Human Pose Annotations," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 6988–6997, 2017.

[40: Rhodin et al. 2018]

Helge Rhodin, Jörg Spörri, Isinsu Katircioglu, Victor Constantin, Frédéric Meyer, Erich Müller, Mathieu Salzmann, and Pascal Fua, "Learning Monocular 3D Human Pose Estimation From Multi-View Images," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 8437–8446, 2018.

doi:10.6342/NTU202202468

[41: Iskakov et al. 2019]
Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov, "Learnable Triangulation of Human Pose," Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7718–7727, 2019.

[42: He et al. 2020]
Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I. Yu, "Epipolar Transformers," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7779–7788, 2020.

[43: Coskun et al. 2017]
Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari, "Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization," Proceedings of the IEEE International Conference on Computer Vision, pp. 5524–5532, 2017.

[44: Katircioglu et al. 2018]
Isinsu Katircioglu, Bugra Tekin, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua, "Learning Latent Representations of 3D Human Pose with Deep Neural Networks," International Journal of Computer Vision, Vol. 126, No. 12, pp. 1326–1341, Dec. 2018.

[45: Tekin et al. 2016]
Bugra Tekin, Artem Rozantsev, Vincent Lepetit, and Pascal Fua, "Direct Prediction of 3D Body Poses from Motion Compensated Sequences," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Jun. 26, 2016.

[46: Cai et al. 2019]
Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann, "Exploiting Spatial-Temporal Relationships for 3D Pose Estimation via Graph Convolutional Networks," Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 2272–2281, 2019.

[47: Huang et al. 2017]
Yinghao Huang, Federica Bogo, Christoph Lassner, Angjoo Kanazawa, Peter V. Gehler, Javier Romero, Ijaz Akhter, and Michael J. Black, "Towards Accurate Marker-Less Human Shape and Pose Estimation over Time," in Proceedings of 2017 International Conference on 3D Vision (3DV), pp. 421–430, Oct. 2017.

[48: Fang et al. 2017]
Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu, "RMPE: Regional Multi-Person Pose Estimation," Proceedings of the IEEE International Conference on Computer Vision, pp. 2334–2343, 2017.

doi:10.6342/NTU202202468

[49: Redmon & Farhadi 2018]

Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," arXiv:1804.02767 [cs], Apr. 2018.

[50: Julier & Uhlmann 2004]

S.J. Julier and J.K. Uhlmann, "Unscented filtering and nonlinear estimation," Proceedings of the IEEE, Vol. 92, No. 3, pp. 401–422, Mar. 2004.

[51: Lin et al. 2015]

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick and Piotr Dollár, "Microsoft COCO: Common Objects in Context," arXiv:1405.0312 [cs], Feb. 2015. [Official website]: https://cocodataset.org/

[52: Gan & Dai 2011]

Yahui Gan and Xianzhong Dai, "Base frame calibration for coordinated industrial robots," Robotics and Autonomous Systems, Vol. 59, No. 7, pp. 563–570, Jul. 2011.

[53: Eberly 2008]

David Eberly. "Euler angle formulas." *Geometric Tools, LLC, Technical Report,* 2008. [Online Available]: https://www.geometrictools.com/Documentation/EulerAngles.pdf (accessed Apr. 19, 2022)

[54: Zhang 2000]

Z. Zhang, "A flexible new technique for camera calibration," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 11, pp. 1330–1334, Nov. 2000.

[55: Ude et al. 2004]

Aleš Ude, Christopher G. Atkeson, and Marcia Riley, "Programming full-body movements for humanoid robots by observation," Robotics and Autonomous Systems, Vol. 47, No. 2, pp. 93–108, Jun. 2004.

[56: Agamennoni et al. 2011]

Gabriel Agamennoni, Juan I. Nieto, and Eduardo M. Nebot, "An outlier-robust Kalman filter," in Proceedings of 2011 IEEE International Conference on Robotics and Automation, pp. 1551–1558, May 2011.

[57: Ting et al. 2007]

Jo-Anne Ting, Evangelos Theodorou, and Stefan Schaal, "A Kalman filter for robust outlier detection," in Proceedings of 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1514–1519, Oct. 2007.

doi:10.6342/NTU202202468

[58: Mu & Yuen 2015]
He-Qing Mu and Ka-Veng Yuen, “Novel Outlier-Resistant Extended Kalman Filter for Robust Online Structural Identification,” Journal of Engineering Mechanics, Vol. 141, No. 1, p. 04014100, Jan. 2015.
[59: Gustafsson 1996]
F. Gustafsson, “Determining the initial states in forward-backward filtering,” IEEE Transactions on Signal Processing, Vol. 44, No. 4, pp. 988–992, Apr. 1996.
[60: Pavllo et al. 2019]
Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli, “3D human pose estimation in video with temporal convolutions and semi-supervised training,” arXiv, arXiv:1811.11742, Mar. 2019. doi: 10.48550/arXiv.1811.11742.
[61: Lin & Lee 2019]
Jiahao Lin and Gim Hee Lee, “Trajectory Space Factorization for Deep Video-Based 3D Human Pose Estimation,” arXiv, arXiv:1908.08289, Aug. 2019. doi: 10.48550/arXiv.1908.08289.
[62: Chen et al. 2022]
Tianlang Chen, Chen Fang, Xiaohui Shen, Yiheng Zhu, Zhili Chen, and Jiebo Luo, “Anatomy-Aware 3D Human Pose Estimation With Bone-Based Pose Decomposition,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 32, No. 1, pp. 198–209, Jan. 2022.
[63: Wang et al. 2020]
Jingbo Wang, Sijie Yan, Yuanjun Xiong, and Dahua Lin, “Motion Guided 3D Pose Estimation from Videos,” arXiv, arXiv:2004.13985, Apr. 2020. doi: 10.48550/arXiv.2004.13985.
[64: Li et al. 2022]
Wenhao Li, Hong Liu, Runwei Ding, Mengyuan Liu, Pichao Wang, and Wenming Yang, “Exploiting Temporal Contexts with Strided Transformer for 3D Human Pose Estimation,” arXiv, arXiv:2103.14304, Jan. 2022. doi: 10.48550/arXiv.2103.14304.
[65: Bouazizi et al. 2021]
Arij Bouazizi, Ulrich Kressel, and Vasileios Belagiannis, “Learning Temporal 3D Human Pose Estimation with Pseudo-Labels,” arXiv, arXiv:2110.07578, Oct. 2021. doi: 10.48550/arXiv.2110.07578.
[66: Qiu et al. 2019]

doi:10.6342/NTU202202468

Haibo Qiu, Chunyu Wang, Jingdong Wang, Naiyan Wang, and Wenjun Zeng, "Cross View Fusion for 3D Human Pose Estimation," arXiv, arXiv:1909.01203, Sep. 2019. doi: 10.48550/arXiv.1909.01203.

[67: Gordon et al. 2022]

Brian Gordon, Sigal Raab, Guy Azov, Raja Giryes, and Daniel Cohen-Or, "FLEX: Extrinsic Parameter-free Multi-view 3D Human Motion Reconstruction," arXiv, arXiv:2105.01937, Mar. 2022. doi: 10.48550/arXiv.2105.01937.

[68: Reddy et al. 2021]

N. Dinesh Reddy, Laurent Guigues, Leonid Pishchulin, Jayan Eledath, and Srinivasa G. Narasimhan, "TesseTrack: End-to-End Learnable Multi-Person Articulated 3D Pose Tracking," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15190–15200, 2021.

[69: Gupta 2020]

Vikas Gupta, "Back to the Future: Joint Aware Temporal Deep Learning 3D Human Pose Estimation," arXiv, arXiv:2002.11251, Feb. 2020. doi: 10.48550/arXiv.2002.11251.

[70: Fang et al. 2018]

Haoshu Fang, Yuanlu Xu, Wenguan Wang, Xiaobai Liu, and Song-Chun Zhu, "Learning Pose Grammar to Encode Human Body Configuration for 3D Pose Estimation." arXiv, Jan. 04, 2018, doi: 10.48550/arXiv.1710.06513

doi:10.6342/NTU202202468

Books:

[71: Hartley & Zisserman 2004]

Richard Hartley and Andrew Zisserman, "Multiple view geometry in computer vision." Cambridge: Cambridge University Press, 2004.

[72: Wan et al. 2004]

Eric A. Wan, Rudolph van der Merwe, and Simon Haykin, "The Unscented Kalman Filter," Kalman Filtering and Neural Networks. John Wiley & Sons, 2004.

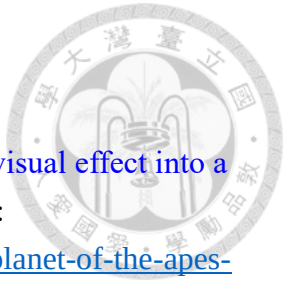
[73: Chatterjee & Hadi 2006]

Samprit Chatterjee and Ali S. Hadi, "Simple Linear Regression," in Regression Analysis by Example, John Wiley & Sons, Ltd, 2006, pp. 21–51.

[74: Anderson & Moore 2007]

Brian D. O. Anderson and John B. Moore, Optimal Control: Linear Quadratic Methods. Courier Corporation, 2007.

doi:10.6342/NTU202202468



Websites:

[75: Bishop 2017]

Bryan Bishop, "How War for the Planet of the Apes turned a visual effect into a reluctant hero," The Verge, July 18, 2017. [Online]. Available: https://www.theverge.com/2017/7/18/15988096/war-for-the-planet-of-the-apes-joe-letteri-visual-effects-interview (accessed on Jan. 24 2022).

[76: Hogan 2020]

Alex Hogan, "Watch: Hollywood motion capture technology finds a new role in hospital rehab," Feb. 12, 2020 [Online]. Available: https://www.statnews.com/2020/02/12/motion-capture-technology-hospitals-physical-rehabilitation/(accessed on Jan. 24 2022)

[77: VICON 2022]

VICON, Inc. "Lower limb monitoring and ACL best practices - Imeasureu revisits some of our recent case studies published with world-class practitioners," Dec. 01, 2022. [Online]. Available: https://www.vicon.com/cms/wp-content/uploads/2022/01/PS4960_IMU_Case_Study.pdf

[78: Dutt 2018]

Arjun Dutt, "Radically Simple: AI Startup Makes 3D Motion Capture a Breeze for All," May 8, 2018. [Online]. Available: https://blogs.nvidia.com/blog/2018/05/08/radical-3d-motion-capture/

[79: Neuron 2022]

Neuron, Perception, Inc. "Perception Neuron Motion Capture | Motion Capture for All." https://neuronmocap.com/ (accessed Feb. 28, 2022).

[80: Xsens 2022]

Xsens, Inc. "Xsens 3D motion tracking." https://www.xsens.com (accessed Feb. 28, 2022).

[81: The Imaging Source 2022]

The Imaging Source, Inc. The Imaging Source. [Online]. Available: https://www.theimagingsource.com/ (accessed Apr. 05, 2022).

[82: Sure Technology Corporation 2022]

Sure Technology Corporation, Inc. MI – CCTV 鏡頭. [Online]. Available: https://www.surevision.com.tw/ (accessed Apr. 05, 2022).

[83: Bradski 2022]

Bradski, G., "The OpenCV Library," Dr. Dobb's Journal of Software Tools. [Online Document]: https://docs.opencv.org. (accessed Apr. 05, 2022).

doi:10.6342/NTU202202468

[84: DSHS.WA 2022]

Washington State Department of Social and Health Services (DSHS.WA), "Range of Joint Motion Evaluation Chart" [Available]: https://www.dshs.wa.gov/sites/default/files/forms/pdf/13-585a.pdf (accessed Mar. 14, 2022).

[85: Jekel 2022]

Charles Jekel, "Least Squares Sphere Fit." https://jekel.me/2015/Least-Squares-Sphere-Fit/ (accessed Apr. 26, 2022).

doi:10.6342/NTU202202468

Appendix A
Optimal rigid body point solution

Assume there are N frames, n points in a rigid body **static** in $\{world\}$. The position of the i-th point **with noise** is denoted as $\boldsymbol{p}_i$. The vector from $\boldsymbol{p}_j$ to $\boldsymbol{p}_i$ at frame t is denoted as $(\boldsymbol{p}_i - \boldsymbol{p}_j)_t$. The optimal rigid body point estimation of the i-th point position is denoted as $\hat{\boldsymbol{p}}_i$.

For a rigid body, the vector between any two points in it should be constant. Therefore, we define the optimization problem to estimate the point positions except $\hat{\boldsymbol{p}}_0$ on the rigid body with minimizing squared sum of the vector differences in different frames as:

$$\min_{\hat{\boldsymbol{p}}_1,\dots,\hat{\boldsymbol{p}}_{n-1}} \sum_{t=0}^{N-1} \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \left(\left\| (\boldsymbol{p}_i - \boldsymbol{p}_j)_t - (\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j) \right\| \right)^2 \tag{A.1}$$

Define a cost function as J:

$$J = \sum_{t=0}^{N-1} \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} \left(\left\| (\boldsymbol{p}_i - \boldsymbol{p}_j)_t - (\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j) \right\| \right)^2 \tag{A.2}$$

While setting $\boldsymbol{d}_{ij,t}$ and $\hat{\boldsymbol{d}}_{ij}$ as:

$$\boldsymbol{d}_{ij,t} = (\boldsymbol{p}_i - \boldsymbol{p}_j)_t \tag{A.3}$$

$$\hat{\boldsymbol{d}}_{ij} = \hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j \tag{A.4}$$

doi:10.6342/NTU202202468

The cost J_t at time t is equal:

$$J_t = \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\left(\left\|(\boldsymbol{p}_i - \boldsymbol{p}_j)_t - (\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j)\right\|\right)^2$$

$$= \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\left(\|\boldsymbol{d}_{ij,t} - \hat{\boldsymbol{d}}_{ij}\|\right)^2$$

$$= \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}(\boldsymbol{d}_{ij,t} - \hat{\boldsymbol{d}}_{ij})^T(\boldsymbol{d}_{ij,t} - \hat{\boldsymbol{d}}_{ij})$$

$$= \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\hat{\boldsymbol{d}}_{ij}{}^T\hat{\boldsymbol{d}}_{ij} - 2\boldsymbol{d}_{ij,t}{}^T\hat{\boldsymbol{d}}_{ij} + \boldsymbol{d}_{ij,t}{}^T\boldsymbol{d}_{ij,t} \qquad \text{(A.5)}$$

$$= \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}(\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j)^T(\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j) - 2\boldsymbol{d}_{ij,t}{}^T(\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j) + \boldsymbol{d}_{ij,t}{}^T\boldsymbol{d}_{ij,t}$$

$$= (n-1)\sum_{i=0}^{n-1}\hat{\boldsymbol{p}}_i{}^T\hat{\boldsymbol{p}}_i - 2\left[\sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}(\hat{\boldsymbol{p}}_i{}^T\hat{\boldsymbol{p}}_j) + \boldsymbol{d}_{ij,t}{}^T(\hat{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_j)\right]$$

$$+ \sum_{i=0}^{n-2}\sum_{j=i+1}^{n-1}\boldsymbol{d}_{ij,t}{}^T\boldsymbol{d}_{ij,t}$$

Then, take the partial derivative for J_t with respect to the estimated point position $\hat{\boldsymbol{p}}_k$, $\forall k = 1, \ldots, n-1$:

$$\frac{\partial J_t}{\partial \hat{\boldsymbol{p}}_k} = 2(n-1)\hat{\boldsymbol{p}}_k - 2\left[\sum_{i=0}^{k-1}(\hat{\boldsymbol{p}}_i - \boldsymbol{d}_{ik,t}) + \sum_{i=k+1}^{n-1}(\hat{\boldsymbol{p}}_i + \boldsymbol{d}_{ki,t})\right] \qquad \text{(A.6)}$$

doi:10.6342/NTU202202468

Next, get the partial derivative for J with summation:

$$\frac{\partial J}{\partial \hat{\boldsymbol{p}}_k} = \frac{\partial (\sum_t J_t)}{\partial \hat{\boldsymbol{p}}_k} = N\left\{2(n-1)\hat{\boldsymbol{p}}_k - 2\left[\sum_{i=0}^{k-1}\left(\hat{\boldsymbol{p}}_i - \bar{\boldsymbol{d}}_{ik}\right) + \sum_{i=k+1}^{n-1}\left(\hat{\boldsymbol{p}}_i + \bar{\boldsymbol{d}}_{ki}\right)\right]\right\} \quad \text{(A.7)}$$

$$\bar{\boldsymbol{d}}_{ik} = \frac{1}{N}\sum_{t=0}^{N-1} \boldsymbol{d}_{ik,t} = -\bar{\boldsymbol{d}}_{ki} \quad \text{(A.8)}$$

To get the optimal $\hat{\boldsymbol{p}}_k$, $\frac{\partial J}{\partial \hat{\boldsymbol{p}}_k} = \mathbf{0}$ should be satisfied. Therefore,

$$N\left\{2(n-1)\hat{\boldsymbol{p}}_k - 2\left[\sum_{i=0}^{k-1}\left(\hat{\boldsymbol{p}}_i - \bar{\boldsymbol{d}}_{ik}\right) + \sum_{i=k+1}^{n-1}\left(\hat{\boldsymbol{p}}_i + \bar{\boldsymbol{d}}_{ki}\right)\right]\right\} = \mathbf{0} \quad \text{(A.9)}$$

$$\Rightarrow (n-1)\hat{\boldsymbol{p}}_k - \left[\sum_{i=0}^{k-1}\left(\hat{\boldsymbol{p}}_i - \bar{\boldsymbol{d}}_{ik}\right) + \sum_{i=k+1}^{n-1}\left(\hat{\boldsymbol{p}}_i + \bar{\boldsymbol{d}}_{ki}\right)\right] = \mathbf{0} \quad \text{(A.10)}$$

$$\Rightarrow n\hat{\boldsymbol{p}}_k - \sum_{i=0}^{n-1}\hat{\boldsymbol{p}}_i + \left(\sum_{i=0}^{k-1}\bar{\boldsymbol{d}}_{ik}\right) - \left(\sum_{i=k+1}^{n-1}\bar{\boldsymbol{d}}_{ki}\right) + \bar{\boldsymbol{d}}_{kk} = \mathbf{0} \quad \text{(A.11)}$$

where $\bar{\boldsymbol{d}}_{kk} = \mathbf{0}$

$$\Rightarrow n\hat{\boldsymbol{p}}_k - \sum_{i=0}^{n-1}\hat{\boldsymbol{p}}_i + \left(\sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{ik}\right) = \mathbf{0} \quad \text{(A.12)}$$

$$\Rightarrow n\hat{\boldsymbol{p}}_k - \sum_{i=0}^{n-1}\hat{\boldsymbol{p}}_i = -\sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{ik} \quad \text{(A.13)}$$

doi:10.6342/NTU202202468

Take the summation of Equation (A.13) for $k = 1, \ldots, n-1$:

$$
\begin{aligned}
n\sum_{i=1}^{n-1}\hat{\boldsymbol{p}}_i - (n-1)\sum_{i=0}^{n-1}\hat{\boldsymbol{p}}_i &= -\sum_{k=1}^{n-1}\sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{ik} \\
&= -\sum_{k=1}^{n-1}\sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{ik} - \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0} + \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0} \\
&= -\sum_{k=0}^{n-1}\sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{ik} + \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0} \\
&= \mathbf{0} + \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0} = \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0}
\end{aligned}
\tag{A.14}
$$

The equation $\sum_{k=0}^{n-1}\sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{ik} = \mathbf{0}$ is because both $\bar{\boldsymbol{d}}_{ab}$ and $\bar{\boldsymbol{d}}_{ba}$ would be added and canceled in the summation. Then, it can be derived as following equation.

$$
\Rightarrow \sum_{i=1}^{n-1}\hat{\boldsymbol{p}}_i - n\hat{\boldsymbol{p}}_0 = \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0} \tag{A.15}
$$

$$
\Rightarrow \sum_{i=1}^{n-1}\hat{\boldsymbol{p}}_i = n\hat{\boldsymbol{p}}_0 + \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0} \tag{A.16}
$$

With Equation (A.13) and Equation (A.16), we get obtain the optimal rigid body point solution as:

$$
\begin{aligned}
\hat{\boldsymbol{p}}_k &= \frac{1}{n}\left(\sum_{i=0}^{n-1}\hat{\boldsymbol{p}}_i - \sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{ik}\right) \\
&= \hat{\boldsymbol{p}}_0 + \frac{1}{n}\left(\sum_{i=0}^{n-1}\bar{\boldsymbol{d}}_{i0} - \bar{\boldsymbol{d}}_{ik}\right) \\
&= \hat{\boldsymbol{p}}_0 + \bar{\boldsymbol{d}}_{k0}
\end{aligned}
\tag{A.17}
$$

while $\hat{\boldsymbol{p}}_0$ is defined as the anchor of the rigid body and can be set arbitrarily.

doi:10.6342/NTU202202468

Appendix B
KKT condition for state constraints with Kalman gain

As the original Kalman gain $\boldsymbol{K}_t$ is determined to minimize the trace of state covariance $\boldsymbol{P}_{t|t}$. Considering the state constraints with lower bound $\boldsymbol{x}_{lower}$ and upper bound $\boldsymbol{x}_{upper}$, the Kalman gain can be obtained by solving Equation (B.1).

$$\min_{\boldsymbol{K}_t} tr(\boldsymbol{P}_{t|t})$$
$$s.t. \boldsymbol{x}_{lower} \leq \hat{\boldsymbol{x}}_t, \ \hat{\boldsymbol{x}}_t \leq \boldsymbol{x}_{upper} \tag{B.1}$$

With the state measurement update equation $\hat{\boldsymbol{x}}_t = \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t$, the inequality constraints can be rewritten as:

$$\begin{cases} \boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t \leq 0 \\ \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper} \leq 0 \end{cases} \tag{B.2}$$

Then, the Lagrangian L is calculated by Equation (B.3) with Lagrange multipliers $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$:

$$L = tr(\boldsymbol{P}_{t|t}) + \boldsymbol{\mu}_1^T(\boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t) + \boldsymbol{\mu}_2^T(\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper}) \tag{B.3}$$

doi:10.6342/NTU202202468

With the update equation of $\boldsymbol{P}_{t|t}$ with arbitrary Kalman gain $\boldsymbol{K}_t$ mentioned in Equation (5.59), L is rewritten as:

$$\begin{aligned} L = tr\left(\boldsymbol{P}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{H}_t\boldsymbol{P}_{t|t-1} - \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t{}^T\boldsymbol{K}_t{}^T + \boldsymbol{K}_t\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}\boldsymbol{K}_t{}^T\right) \\ + \boldsymbol{\mu}_1^T\left(\boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t\right) \\ + \boldsymbol{\mu}_2^T\left(\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper}\right) \end{aligned} \tag{B.4}$$

Taking the partial derivative with respect to $\boldsymbol{K}_t$ equal to zero matrix:

$$\frac{\partial L}{\partial \boldsymbol{K}_t} = \boldsymbol{0} \Rightarrow -2\left(\boldsymbol{H}_t\boldsymbol{P}_{t|t-1}\right)^T + 2\boldsymbol{K}_t\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1} + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\boldsymbol{\varepsilon}_t^T = \boldsymbol{0} \tag{B.5}$$

Then, the Kalman gain with KKT condition is calculated as Equation (B.6) with the condition in Equation (B.7) and Equation (B.8):

$$\boldsymbol{K}_t = \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}^{-1} + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\boldsymbol{\varepsilon}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}^{-1} \tag{B.6}$$

$$\begin{cases} \boldsymbol{\mu}_1 \geq \boldsymbol{0} \\ \boldsymbol{\mu}_2 \geq \boldsymbol{0} \end{cases}, \quad \begin{cases} \boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t \leq \boldsymbol{0} \\ \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper} \leq \boldsymbol{0} \end{cases} \tag{B.7}$$

$$\begin{cases} \mu_{1,i}\left(\boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t\right)_i = 0 \\ \mu_{2,i}\left(\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper}\right)_i = 0 \end{cases} \forall i \tag{B.8}$$

where $\mu_{1,i}$ is the i-th element in $\boldsymbol{\mu}_1$, $\mu_{2,i}$ is the i-th element in $\boldsymbol{\mu}_2$, and $(\cdot)_i$ is the i-th element in the vector.

doi:10.6342/NTU202202468

With Equation (B.6), we can write the following equation:

$$\boldsymbol{a}_t = \boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}^{-1}\boldsymbol{\varepsilon}_t \tag{B.9}$$

$$\boldsymbol{b}_t = \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{x}_{upper} + \boldsymbol{P}_{t|t-1}\boldsymbol{H}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}^{-1}\boldsymbol{\varepsilon}_t \tag{B.10}$$

$$c_t = \boldsymbol{\varepsilon}_t^T\boldsymbol{P}_{\hat{\boldsymbol{z}},t|t-1}^{-1}\boldsymbol{\varepsilon}_t \geq 0 \in \mathbb{R} \tag{B.11}$$

$$\begin{cases} \boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t = \boldsymbol{a}_t - c_t(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ \hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper} = \boldsymbol{b}_t + c_t(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \end{cases} \tag{B.12}$$

Therefore, Equation (B.8) can be rewritten as:

$$\begin{cases} \mu_{1,i}\left(\boldsymbol{a}_t - c_t(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right)_i = 0 \\ \mu_{2,i}\left(\boldsymbol{b}_t + c_t(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)\right)_i = 0 \end{cases} \forall i \tag{B.13}$$

$$\Rightarrow \begin{cases} \mu_{1,i}\left[a_{t,i} - c_t(\mu_{1,i} - \mu_{2,i})\right] = 0 \\ \mu_{2,i}\left[b_{t,i} + c_t(\mu_{1,i} - \mu_{2,i})\right] = 0 \end{cases} \forall i \tag{B.14}$$

$$\Rightarrow \begin{cases} \mu_{1,i} = 0 \quad or \quad a_{t,i} - c_t(\mu_{1,i} - \mu_{2,i}) = 0 \\ \mu_{2,i} = 0 \quad or \quad b_{t,i} + c_t(\mu_{1,i} - \mu_{2,i}) = 0 \end{cases} \forall i \tag{B.15}$$

There are 4 situations determined by if $\mu_{1,i}$ equal 0 or not and $\mu_{2,i}$ equal 0 or not.

First, we consider the situation $a_{t,i} - c_t(\mu_{1,i} - \mu_{2,i}) = 0,\ \mu_{2,i} = 0$. In the situation, there are two condition to check for Equation (B.7):

$$\begin{cases} \mu_{1,i} = \dfrac{1}{c_t}a_{t,i} \geq 0 \\ \left(\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper}\right)_i = b_{t,i} + c_t\mu_{1,i} = x_{lower,i} - x_{upper,i} \leq 0 \end{cases} \tag{B.16}$$

The second condition is trivial. Since c_t is always nonnegative, the situation is satisfied if $a_{t,i} \geq 0$.

doi:10.6342/NTU202202468

Next, we consider the situation $\mu_{1,i} = 0,\ \mu_{2,i} = 0$. In the situation, there are two condition to check for Equation (B.7):

$$\begin{cases} \left(\boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t\right)_i = a_{t,i} \le 0 \\ \left(\hat{\boldsymbol{x}}_{t|t-1} + \boldsymbol{K}_t\boldsymbol{\varepsilon}_t - \boldsymbol{x}_{upper}\right)_i = b_{t,i} \le 0 \end{cases} \tag{B.17}$$

Therefore, if $a_{t,i} \le 0$ and $b_{t,i} \le 0$, the situation will be satisfied.

Third, we consider the situation $\mu_{1,i} = 0,\ b_{t,i} + c_t(\mu_{1,i} - \mu_{2,i}) = 0$. In the situation, there are two condition to check for Equation (B.7):

$$\begin{cases} \left(\boldsymbol{x}_{lower} - \hat{\boldsymbol{x}}_{t|t-1} - \boldsymbol{K}_t\boldsymbol{\varepsilon}_t\right)_i = a_{t,i} + c_t\mu_{2,i} = x_{lower,i} - x_{upper,i} \le 0 \\ \mu_{2,i} = \dfrac{1}{c_t} b_{t,i} \ge 0 \end{cases} \tag{B.18}$$

Because the first condition is trivial, if $b_{t,i} \ge 0$, this situation can be satisfied.

Last, for the situation $a_{t,i} - c_t(\mu_{1,i} - \mu_{2,i}) = 0$, $b_{t,i} + c_t(\mu_{1,i} - \mu_{2,i}) = 0$, the upper bound and lower bound coincide as Equation (B.19) shown:

$$\begin{aligned} 0 = &\ a_{t,i} - c_t(\mu_{1,i} - \mu_{2,i}) + b_{t,i} + c_t(\mu_{1,i} - \mu_{2,i}) \\ = &\ a_{t,i} + b_{t,i} \\ = &\ \left(\boldsymbol{x}_{lower} - \boldsymbol{x}_{upper}\right)_i \end{aligned} \tag{B.19}$$

For the situation, the element $(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)_i$ used to calculate Kalman gain is:

$$\begin{aligned} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)_i &= \mu_{1,i} - \mu_{2,i} \\ &= \frac{1}{c_t} a_{t,i} \\ &= -\frac{1}{c_t} b_{t,i} \end{aligned} \tag{B.20}$$

doi:10.6342/NTU202202468